1

AD A118134

DTIC FILE COPY

UNITED STATES AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

82 08 11 048

①

AFIT/GE/EE/81D-48

DESIGN AND ANALYSIS OF A MULTIVARIABLE, DIGITAL CONTROLLER FOR THE A-7D DIGITAC II AIRCRAFT AND THE DEVELOPMENT OF AN INTERACTIVE COMPUTER DESIGN PROGRAM

THESIS

AFIT/GE/EE/81D-48 Douglas S. Porter
Capt USAF

AFIT/GE/EE/81D-48

DESIGN AND ANALYSIS OF A MULTIVARIABLE, DIGITAL CONTROLLER FOR THE A-7D DIGITAC II AIRCRAFT AND THE DEVELOPMENT OF AN INTERACTIVE COMPUTER DESIGN PROGRAM

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

Douglas S. Porter, B. S.

Capt USAF

Graduate Engineering

December 1981

Accession For		
NTIS GRA&I		✓
DTIC TAB		☐
Unannounced		☐
Justification		
By		
Distribution/		
Availability Codes		
Dist	Avail and/or Special	
A		

Preface

This thesis topic was suggested and sponsored by Duane P. Robertus of the Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio. The study gave me much insight into the application of course work at the Air Force Institute of Technology in aircraft dynamics, aircraft flight control, and computer-aided design programs. It was refreshing to be acquainted with up-and-coming multivariable design techniques for aircraft controller design and to be able to generate a design package for future Air Force Institue of Technology students and Wright-Patterson AFB control engineers, who may be involved in this field.

This thesis is the composite of unmeasurable hours of work and was accomplished in association with two fellow Air Force Institute of Technology students. Lt. Randy N. Paschall was paramount in the development of the aircraft control derivatives and Lt. Joe S. Smyth assisted in the initial development of the computer package. Their associate theses make use of the computer-aided design program.

It would not have been possible to attempt this thesis without the cooperation of Professor Brian Porter who is developing multivariable design theory, under USAF contract, at the University of Salford, England. My unending thanks is also expressed to Professors Constantine H. Houpis and John J. D'Azzo of the Department of Electrical Engineering at the Air Force Institute of Technology, who gave me countless hours of their time during the thesis development and suggested many alternatives which guided the thesis progress and writing. I also wish to acknowledge committee member, Capt. James T. Silverthorn, Department of

Aeronautics and Astronautics, Air Force Institute of Technology, for his suggestions in the development of the aircraft equations of motion and stability derivatives.

I am especially thankful to Air Force Institute of Technology student, Lt. Eileen W. Bjorkman, who suggested several corrections to the many versions of the computer program. Acknowledgement is also tendered to the personnel of the ASD Computer Open Shop, Wright-Patterson AFB, Ohio.

Finally, I wish to thank my fiance, Lt. Brenda F. Stone, for her patience and help during the production of this thesis. And to my family and friends, I wish to express my gratitude for their understanding.

Capt. Douglas S. Porter

Contents

Contents

List of Figures

List of Figures

List of Tables

Abstract

This thesis investigates a singular perturbation method in the design of a multivariable, error-actuated digital tracking controller for the Mach 0.18 flight condition of the A-7D Digitac II aircraft. The study includes the development of a computer-aided design package to accomplish the control law design and simulation.

The aircraft model is taken from an associate thesis and is a six degree-of-freedom aircraft model that contains separate left and right horizontal stabilator control surfaces. Thus, the equations of motion cannot be decoupled for the design of the control laws.

The complete design process used to obtain a tracker control law is presented. The presentation includes a discussion of the robustness of the control law when it is used for the same aircraft at various flight conditions. The effect on aircraft stability is also shown in the event of the failure of the right horizontal stabilator.

The thesis concludes that tracker control laws can be designed for this aircraft model of interest, but additional modification of the control law is required to satisfy the robustness and reconfiguration requirements. Also, a more accurate aircraft model should be developed to provide further analysis.

The computer design and simulation package is available for use by the engineering community.

DESIGN AND ANALYSIS OF A MULTIVARIABLE, DIGITAL CONTROLLER FOR THE A-7D DIGITAC II AIRCRAFT AND THE DEVELOPMENT OF AN INTERACTIVE COMPUTER DESIGN PROGRAM

I Introduction

Many modern aircraft make use of small, but sophisticated, digital computers. One very important application is in the area of computer-generated signals which move the aircraft's primary flight control surfaces. As a result, today's newest aircraft are designed with flight control surfaces which are ultimately moved by electrical signals rather than by mechanical linkages. In these aircraft, the pilot does not have direct control of the aircraft. The aircraft unit involved is called a digital flight control system (DFCS).

The DFCS is programmed with the control laws necessary to direct the desired aircraft surface movements. The control laws require satisfactory operation of each element in the control loop. Redundancy schemes are used to ensure continued operation of the surfaces during reasonable system failures. If, however, a primary flight control surface becomes inoperative or missing (due to combat damage, structural fatigue, etc.), the control laws are invalid and the flight control system cannot be used to properly control the aircraft. The controllability problem is increased if the aircraft is statically unstable, because the aircraft may quickly increase in pitch and bank (Ref 1).

This thesis addresses the development of control laws for a DFCS.

The first chapter presents the associated background, problem statement, assumptions, approach, and sequence of presentation.

Background

There have been many steps in aircraft flight control system development. A major step occurred when analog flight control systems were replaced with DFCSs. The result was faster reaction to input disturbances and increased flexibility in control surface utilization (Ref 2). Past advancements in digital technology have not only improved the computational speed, but lowered the cost for digital computers used in flight control systems.

The A-7D Digitac II aircraft is used to test experimental DFCSs. The test program provides a technological base for future developments of digital and multimode flight control functions (Ref 3:2). Current programs involve DFCSs with detection and isolation capabilities in the event of the failure or loss of an aircraft control surface.

In the event of a severe flight control surface failure, the closed-loop flight control system should be designed to detect the surface failure and direct the remaining flight control surface movements so as to restore aircraft stability and control to an acceptable level. A controllable aircraft is required so the pilot can recover the aircraft to a safe ejection attitude or fly the aircraft to a safe landing site.

The required type of DFCS is based on reconfigurable, multivariable tracker control laws. Reconfiguration implies changing from normal operation to system operation based on control laws that account for inoperative surfaces. The term "tracker" means that flight control surface movements (outputs) follow pilot stick deflections (inputs). The word "multivariable" denotes that there are multiple inputs and outputs.

Problem

The Air Force Flight Dynamics Laboratory is presently investigating digital, reconfigurable, multivariable tracker control laws and their applications. A topic of interest is the development of a robust control law which accommodates surface failures and can operate in an aircraft's entire range of flight conditions. The control law should also be easy to develop.

Consider the example of a failure or loss of the right aileron of an aircraft. In this case, additional forces and moments are generated that are not normally present. Continued aileron commands (to the left aileron) would produce both longitudinal forces and lateral-directional commands. A multivariable control law will redistribute the aileron control signals to other primary aircraft surfaces so that the aircraft remains controllable. Of course, as a first requirement, the control law must operate under normal conditions and over a wide range of flight conditions.

Assumptions

Lt. David W. Potts developed stability derivatives and an aircraft model for the A-7D aircraft in a previous thesis (Ref 3). The A-7D Digitac II aircraft model is used in this study and is assumed to be valid. His work assumes that: (1) flight control surfaces can be divided into independant surfaces (left aileron, right aileron, etc.)

during analysis of aircraft forces and moments, (2) the aircraft makes only small perturbations about a trimmed, straight and level flight condition, and (3) the aircraft equations of motion are developed using aircraft stability axes (Ref 3:4). These assumptions, which are retained, are needed to reduce the complexity of the aircraft system analysis.

Approach

The object of this thesis is to investigate the development of a digital, reconfigurable, multivariable tracker control law for the A-7D Digitac II aircraft by applying the design techniques proposed by Professor Brian Porter, University of Salford, England (Refs 4; 5; and 6). Professor Porter is currently under U. S. Government contract continuing development in multivariable control law theory. The study of this thesis runs parallel to the thesis development by Lt. Randy N. Paschall (Ref 7), in that the aircraft control derivatives, for the Mach 0.18, 0.6, and 0.8 flight conditions, are developed conjunctively.

Specifically, this thesis presents the development of a control law for the Mach 0.18 flight condition. In addition, the presentation supplies an interactive computer program for the design and simulation of digital, multivariable control laws using Professor Porter's techniques.

Sequence of Presentation

This thesis contains seven chapters. Chapter II develops the aircraft model and references Lt. Paschall's thesis for the aircraft equations of motion and stability derivatives. Chapter III summarizes

Professor Porter's multivariable control law theory. The associated computer program, MULTI, is introduced in Chapter IV, which relates the general development of the program; Chapter V discusses the program's structure. The control law design for the Mach 0.18 flight condition is given in Chapter VI. Chapter VII provides the results and conclusions of the thesis, suggests further projects related to the design methods, and presents possible modifications and/or additions to MULTI.

II A7-D Aircraft Models

Introduction

The models for this thesis are based upon the A-7D aircraft model developed by Lt. David W. Potts (Ref 3). The Potts model was modified to account for changes suggested by thesis committee member, Capt. James T. Silverthorn of the Department of Aeronautics and Astronautics at the Air Force Institute of Technology. The aircraft model has six degrees of freedom (D-O-F) and has controls which are divided into separate left and right surfaces. As an example, although the horizontal stabilizer is normally considered as a single control surface, the model of this thesis identifies it as divided into a left horizontal stabilizer and a right horizontal stabilizer. The non-linear six D-O-F aircraft equations of motion are linearized about a nominal operating point to provide linear aircraft equations for the study. Three models are developed which include a landing configuration of 0.18 Mach at 2,000 feet, a medium altitude cruise of 0.6 Mach at 15,000 feet, and a high altitude cruise of 0.8 Mach at 35,000 feet. The opportunity to check for control law robustness is provided by developing aircraft models for all three flight conditions.

This chapter overviews the A-7D aircraft and presents a discussion on the linearized aircraft equations of motion.

Aircraft Description

As a single seat, light attack, fighter aircraft, the A-7D has moderately swept wings and tail surfaces, is designed with a lower fuselage engine inlet duct, and is powered by a TF41-A-1 engine (Ref 3:7). Other pertinant aerodynamic data, as found in Reference 3, is as follows:

Characteristic	Data	Units
Wing area	375	sq. ft.
Wing m.g.c.	10.8	feet
Horizontal Slab area	56.2	sq. ft.
Slab m.g.c.	6.1	feet
Distance from 25% wing m.g.c. to 25% slab m.g.c.	16.2	feet
Cruise weight	25,238	pounds

The abbreviation, m.g.c., stands for mean geometric chord. Further aerodynamic data for the A-7D Digitac aircraft can be found in Reference 8.

System Models

The approach for system model development given in the Potts reference is used to develop the three aircraft models of this thesis. A review of the techniques involved to develop the aircraft equations of motion and the control derivatives for this thesis is given in Reference 7. This work also contains the state equation matrices of all three models and a discussion on possible actuators and sensors for the system.

Aircraft Equations. The aircraft equations of motion are developed so that a longitudinal control surface can provide lateral motion and a lateral control surface can provide longitudinal motion. This is the requirement to be met when considering the development of a reconfigurable control law for the possibility of a primary control surface failure. The aircraft is modeled with eight inputs: left and right ailerons, left and right spoilers, left and right horizontal stabilizers, a rudder, and flaps.

The aircraft has eight system outputs which are defined as:

Output	Symbol
Forward velocity	u
Flight path angle	γ
Pitch rate	q
Pitch angle	θ
Roll rate	p
Roll angle	ϕ
Yaw rate	r
Sideslip angle	β

Control Derivatives. The non-dimensional control derivatives for the aircraft models were derived with the aid of the Digital DATCOM computer package (Ref 9) and from aircraft data charts found in Ref. 10 and Ref. 11. To learn the method for developing the control derivatives from the DATCOM and chart data, the Mach 0.6 derivatives were calculated and checked against those found in Ref. 3. The control derivatives were adjusted as necessary, and then the derivatives for the Mach 0.18 and Mach 0.8 flight conditions were computed.

Summary

This chapter begins with a brief discussion of the A-7D Digitac II aircraft. The model's aircraft equations of motion are then described and reference is made to the eight inputs and outputs of the system.

The chapter concludes with a discussion of how the non-dimensional control derivatives are derived.

With the aircraft models for the 0.18, 0.6, and 0.8 Mach flight conditions available, the discussion now turns to the theory used to develop a control law for the model.

III Multivariable Digital Control Law Theory

Introduction

The digital control law methods used in this thesis are those developed by Professor Brian Porter, University of Salford, England. A summary of the digital techniques presented in Refs. 4, 5, and 6, is given in this chapter. These techniques produce a proportional plus integral control signal which is piece-wise constant for each sampling period. Thus, the total system consists of a continuous-time plant and a digital controller.

The continuous state and output equations are defined as:

$$\underline{\dot{x}}(t) = \underline{A}\,\underline{x}(t) + \underline{B}\,\underline{u}(t) + \underline{D}\,\underline{d}(t) \tag{1}$$

$$\underline{y}(t) = \underline{C}\,\underline{x}(t) \tag{2}$$

where

$\underline{A}$ = continuous-time plant matrix

$\underline{B}$ = continuous-time control matrix

$\underline{D}$ = continuous-time disturbance matrix

$\underline{C}$ = continuous-time output matrix

This chapter defines the number of states, inputs, and outputs as n, m, and p, respectively.

There are three design procedures based on the type of plant involved in the design. The type of plants are described as unknown, regular, and irregular. The discrete-time state and output equations of the plants follow the form:

$$\underline{x}(kT + T) = \phi \underline{x}(kT) + \psi \underline{u}(kT) + \Delta \underline{d}(kT) \tag{3}$$

$$\underline{y}(kT) = \Gamma \underline{x}(kT) \tag{4}$$

where

ϕ = sampled-data plant matrix

ψ = sampled-data control matrix

Δ = sampled-data disturbance matrix

Γ = sampled-data output matrix

and

$$\phi = \exp(\underline{A}T) \tag{5}$$

$$\psi = \int_0^T \exp(\underline{A}t)\, \underline{B}\, dt \tag{6}$$

$$\Delta = \int_0^T \exp(\underline{A}t)\, \underline{D}\, dt \tag{7}$$

$$\Gamma = \underline{C} \tag{8}$$

The final section of the chapter contains a description of the range spaces for the matrices and vectors of this chapter's equations.

Unknown Plants (Ref 4)

If the physical system of interest is not already described by a state equation model, it may be costly and time consuming to develop a detailed model. If the steady-state transfer function matrix, $\underline{G}(0)$, is developed for the plant, an error-actuated controller can be designed for the system using the unknown plant technique. The $\underline{G}(0)$ matrix can easily be determined from "off-line" tests for a stable plant. If the state equations are known, the unknown plant method can be used to develop a controller after obtaining the $\underline{G}(0)$ matrix from:

$$\underline{G}(0) = -[\,\underline{C}\,\underline{A}^{-1}\,\underline{B}\,] \tag{9}$$

Stability is assured for this method of controller design if the rank of the steady-state transfer function matrix is equal to the number of system outputs. The system must also have at least as many inputs as outputs. It is assumed that the eigenvalues of the open-loop plant matrix, $\underline{A}$, lie in the stable, left-half s-plane region.

The control law for the error-actuated digital controller incorporates a proportional plus integral control represented by:

$$\underline{u}\,(kT) = T\,[\,\alpha\,\epsilon\,\underline{K}\,e\,(kT) + \epsilon\,\underline{K}\,\underline{z}\,(kT)\,] \tag{10}$$

where

$$\underline{e}\,(kT) = \underline{v}\,(kT) - \underline{y}\,(kT) \tag{11}$$

$$\underline{z}\,(kT + T) = \underline{z}\,(kT) + T\,\underline{e}\,(kT) \tag{12}$$

The parameter, α , determines the proportion of integral of error feedback, $\underline{z}$ (kT), to direct error feedback, $\underline{e}$ (kT); $\underline{v}$ (kT) is the command input vector; T is the sampling time; ϵ is a scalar multiplier. The values of α , ϵ , T, and $\underline{K}$ are chosen such that the eigenvalues of the closed-loop plant matrix are inside the z-plane unit circle. Consequently, in the limit, when k approaches infinity, $\underline{e}\,(kT) \rightarrow 0$, so that, for constant commands, tracking and disturbance rejection is accomplished.

The controller matrix, $\underline{K}$, is found from the relation:

$$\underline{K} = \underline{G}(0)^T\,[\,\underline{G}(0)\,\underline{G}(0)^T\,]^{-1}\Sigma \tag{13}$$

where Σ is a weighting matrix of scalar diagonal elements, σ_i . The controller for the unknown plants can be condensed into the formula:

$$\underline{u}\,(kT) = T\,[\,\underline{K}_0\,\underline{e}\,(kT) + \underline{K}_1\,\underline{z}\,(kT)\,] \tag{14}$$

where

$$\underline{K}_0 = \alpha\,\epsilon\,\underline{K} \tag{15}$$

$$\underline{K}_1 = \epsilon\,\underline{K} \tag{16}$$

Regular Plants (Ref 5)

The state and output equations described in Eqs. (1) thru (8) are also used in this design method. However, the state equation matrices of the plant in the continuous-time domain are partitioned into the form:

$$\underline{A} = \begin{bmatrix} \underline{A}_{11} , \underline{A}_{12} \\ \underline{A}_{21} , \underline{A}_{22} \end{bmatrix} \qquad (17)$$

$$\underline{B} = \begin{bmatrix} 0 \\ \underline{B} \end{bmatrix} \qquad (18)$$

$$\underline{C} = [\ \underline{C}_1 , \underline{C}_2\ ] \qquad (19)$$

For a plant to conform to the regular plant design technique of controller design, the first Markov parameter, $[\underline{C}_2 \underline{B}_2]$, must have rank equal to the number of outputs. The control law is:

$$\underline{u}\,(kT) = (1/T)\,[\,\underline{K}_0\,\underline{e}\,(kT) + \underline{K}_1\,\underline{z}\,(kT)\,] \qquad (20)$$

where Eqs. (15) and (16) apply.

Increasingly better tracking and disturbance rejection occurs as the sampling frequency, (1/T), approaches infinity. The theory associated with this design shows that the transfer function matrix, $\underline{G}(\lambda)$, includes "slow" modes and "fast" modes. The system transmission zeros are included in the "slow" modes and are calculated from the equation:

$$\underline{Z}(\lambda_t) = \left\{ \left| \lambda \underline{I}_{n-p} - \underline{I}_{n-p} - T\,\underline{A}_{11} + T\,\underline{A}_{12}\,\underline{C}_2^{-1}\,\underline{C}_1 \right| = 0 \right\} \qquad (21)$$

The matrix, $\underline{I}$, is the identity matrix. It is essential that the transmission zeros lie inside the unit circle of the z-plane, for the sampling period chosen, so that the system remains stable.

The controller's tracking and disturbance rejection properties are attained if the controller matrix, $\underline{K}$, is designed such that:

$$\underline{K} = (1/\alpha\epsilon)[\underline{C}\,\underline{B}]^{-1}\Sigma \tag{22}$$

Considering the partitions of matrices, $\underline{B}$ and $\underline{C}$, and the definition of $\underline{K}_o$ in Eq. (15), Eq. (22) can be written:

$$\underline{K}_o = [\underline{C}_2\,\underline{B}_2]^{-1}\Sigma \tag{23}$$

Irregular Plants (Ref 6)

The state, output and control law equations in the regular plant design method are also applicable to this section. However, for plants in which the $\underline{C}_2$ matrix is rank deficient, it is necessary to modify the error vector, $\underline{e}$ (kT), so that the first Markov parameter has full rank. Note that the plant control matrix, $\underline{B}$, has the form shown in Eq. (18) and is assumed to have full rank.

The error vector of Eq. (11) is replaced by the following equation:

$$\underline{e}(kT) = \underline{v}(kT) - \underline{w}(kT) \tag{24}$$

where

$$\underline{w}(kT) = [\underline{F}_1, \underline{F}_2]\,\underline{x}(kT) \tag{25}$$

and

$$\underline{F}_1 = [\underline{C}_1 + \underline{M}\,\underline{A}_{11}] \tag{26}$$

$$\underline{F}_2 = [\underline{C}_2 + \underline{M}\,\underline{A}_{12}] \tag{27}$$

The matrix, $\underline{M}$, is a measurement matrix which must be chosen so that the rank of $\underline{F}_2$ is equal to the number of outputs. With constant inputs, the steady state values of both $\underline{w}$ (kT) and $\underline{y}$ (kT) equal $\underline{v}$ (kT), so the system tracks the commanded input $\underline{v}$ (kT) as desired.

The control law is the same as that given in Eq. (20). However, because of the modification of the error vector, the transmission zeros are found from:

$$\underline{Z}(\lambda_t) = \left\{ \left| \lambda \underline{I}_{n-p} - \underline{I}_{n-p} - T \underline{A}_{11} + T \underline{A}_{12} \underline{F}_2^{-1} \underline{F}_1 \right| = 0 \right\} \tag{28}$$

These transmission zeros must lie inside the unit circle of the z-plane.

For irregular plants the controller matrix, $\underline{K}$, is defined as:

$$\underline{K} = (1/\alpha\epsilon) [\underline{F}_2 \underline{B}_2]^{-1} \Sigma \tag{29}$$

Vector/Matrix Range Spaces

This section lists the range spaces for the vectors and matrices of Eqs. (1) to (29). Beginning with the definition that:

n	=	number of states
m	=	number of inputs
p	=	number of outputs

for all designs, $\underline{x}(kT) \in R^n$, $\underline{u}(kT) \in R^m$, $\underline{y}(kT) \in R^p$, $\underline{e}(kT) \in R^p$, $\underline{v}(kT) \in R^p$, $\underline{z}(kT) \in R^p$, $\underline{K} \in R^{m \times p}$, $\underline{K}_0 \in R^{p \times p}$, $\underline{K}_1 \in R^{p \times p}$, and $\Sigma \in R^p$. For both regular and irregular plants, the systems are assumed to be square, that is, m = p. The additional range space definitions are shown in Table I.

Summary

This chapter presents the three digital control law methods developed by Professor Brian Porter. Control laws for unknown, regular,

TABLE I

Definition of Vector/Matrix Range Spaces

REGULAR PLANTS		IRREGULAR PLANTS	
Vector/Matrix	Space	Vector/Matrix	Space
$\underline{x}_1(kT)$	R^{n-p}	$\underline{F}_1$	$R^{p \times (n-p)}$
$\underline{x}_2(kT)$	R^{p}	$\underline{F}_2$	$R^{p \times p}$
$\underline{A}_{11}$	$R^{(n-p) \times (n-p)}$	$\underline{M}$	$R^{p \times (n-p)}$
$\underline{A}_{12}$	$R^{(n-p) \times p}$		
$\underline{A}_{21}$	$R^{p \times (n-p)}$		
$\underline{A}_{22}$	$R^{p \times p}$		
$\underline{B}_2$	$R^{p \times p}$		
$\underline{C}_1$	$R^{p \times (n-p)}$		
$\underline{C}_2$	$R^{p \times p}$		

and irregular plants are implemented in the computer program, MULTI.

Aspects concerning the development of the computer program are given in the next chapter.

IV Development of the MULTI Computer Program

Introduction

MULTI, an interactive, user-oriented computer program, is developed to design and simulate the control laws for unknown, regular, and irregular plants. The computer program includes nearly 2,400 lines of FORTRAN V code and is a full extension of the simulation package for unknown plants, PAK200 (Ref 12). PAK200 was received from the University of Salford, Department of Aeronautical and Mechanical Engineering.

This chapter relates the requirements that are set for the program, the program design emphasis, and the constraints and problems that are faced in the development of MULTI. The next chapter describes the actual program structure of MULTI. Appendix A consists of a User's Manual for MULTI and Appendix B contains a Programmer's Manual that will facilitate future expansion of MULTI.

Program Requirements

The requirements for the computer program are set so that MULTI can take full advantage of the techniques used in designing discrete-time tracking systems for linear multivariable (multiple input, multiple output) plants.

These requirements include:

(1) The computer package must be fully interactive, be user-oriented, and allow for an iterative design process.

(2) The program must be able to detect input errors and supply the user with the cause of the errors.

(3) The program must allow the user to input data from the terminal keyboard or a data file and store pertinent data in local files upon normal program termination.

(4) The package should include design ability for unknown, regular, and irregular plants.

(5) The package must be able to recover when singular matrices are formed and direct the user to apply an alternate design technique.

(6) The package must be able to form or accept a measurement matrix from terminal input and allow the user to perform simple row and column operations on any matrix.

(7) The package must include a simulation of the continuous plant with piece-wise constant control signals, which are generated by a discrete controller, in order to evaluate the control laws developed.

(8) The package must include a plotting selection for quick sketches at the user's terminal or hard-copy CALCOMP plots.

From the requirements above, a structured chart is developed that is shown in Fig. 1. This chart illustrates the modules of the MULTI program and the interconnection between the modules.

Program Design Emphasis

When a computer program is divided into several modules, the content of each module and the interconnections between the modules can significantly affect the operation and complexity of the resulting

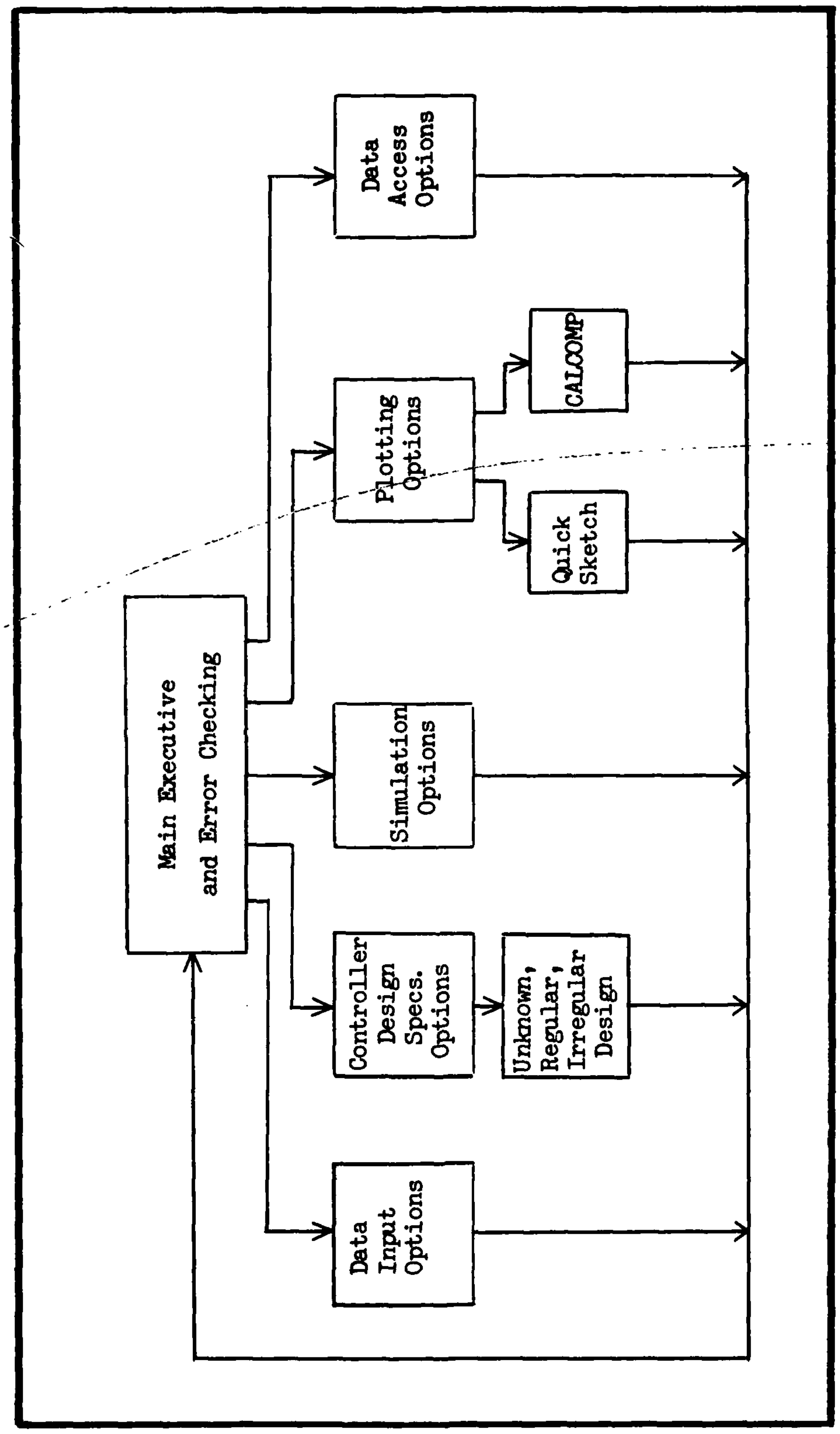

Main Executive and Error Checking
Data Input Options
Controller Design Specs. Options
Unknown, Regular, Irregular Design
Simulation Options
Plotting Options
CALCOMP
Quick Sketch
Data Access Options

Fig. 1 MULTI Structure Chart

program. Coupling and cohesion (Ref 13) are two qualities that are used to measure the overall design. These two concepts form the basis of structured design theory.

Coupling is the measure of the strength of the interconnections between one module and another. It is essential that coupling be kept at a minimum so that changes in one module do not severely affect the other modules. When coupling is retained at a low level, the ease of finding and correcting program bugs is enhanced and the program's life is increased since quick modifications are made possible. In order to keep the level of coupling low, the use of global data is kept at a minimum. When global storage is required, labeled common blocks are used rather than blank common blocks so that the level of coupling is reduced.

Cohesion is the degree of functional relatedness of processing elements within a single module. It also has a direct effect on the ability of a programmer to maintain and modify a program. A high level of cohesion is desirable and is achieved by putting only functionally related processes within each module. However, when memory reduction can be accomplished with a slight loss of cohesion, the loss of cohesion is considered acceptable.

In MULTI, the major functions of the program are directed to lower-level modules. Every module is designed to use a minimum of memory core, and every labeled common block is designed to hold only those elements needed for a specific purpose. Through this type of design, MULTI is able to use minimum computer memory core, have minimum coupling, and retain a high level of cohesion.

Design Constraints

There are two main design constraints when developing a computer program which is as complex as MULTI: one, the computer language that must be used, and two, the availability of computer core memory that is allowed when using an INTERCOM system.

FORTRAN V is chosen as the language for MULTI since AFIT's current programming classes utilize this FORTRAN version and since it is the latest version of FORTRAN available on the ASD CYBER computer system. Using FORTRAN V does not restrict the use of the ASD Library subroutines, even though they are coded in FORTRAN IV, since all are readily compiled into the more powerful version.

Current interactive computer users are constrained to operate in $65{,}000_8$ words of core memory, or less. Therefore, MULTI is defined to operate within this restriction. The core memory restriction is met through the modular design of the program.

Generalized Matrix Inverse

There are matrix inversions occurring at several points during MULTI's operation. It is found that many plant matrices, or combinations of these matrices, are singular and cannot be inverted. An early version of MULTI included a subroutine called GENINV which could be used in such cases.

Subroutine GENINV could be used to find a generalized matrix inverse for a matrix which is singular. Its primary tool for the inversion was built upon the IMSL Library subroutine LGINF (Ref 14).

After a generalized inverse was formed, a computational check was made for accuracy comparisons. It was found that the generalized matrix inversion is not an acceptable solution to the problem of singular matrix inversion. Thus, the GENINV subroutine was discarded.

When singular matrices which must be inverted are encountered, the current MULTI package notifies the user of the problem. The user is then required to augment the plant matrices in some manner, using the synthesis method for irregular plants, to eliminate the problem.

Developmental Problems

The starting point for the initial MULTI program design is the computer package called PAK200 which was written for controller design simulations at the University of Salford, England. The object in developing MULTI, a similar computer package, is to make PAK200 compatible with the ASD CYBER computer, i.e. certain computational subroutines from the ASD Subroutine Library are substituted or incorporated into PAK200.

The final MULTI computer package that is now available is a completely interactive design and simulation tool for all three types of plants (unknown, regular, and irregular). The main body of the code from PAK200 is entirely contained in a part of one option of MULTI. Two subroutines, dealing with the differential equation formation and the value of the output at the end of each time increment, are also retained to provide compatability with the original simulation code.

The substitution of the ASD Library subroutines into PAK200 created many development problems in the formulation of the final MULTI package. Several other problems were also encountered as related below.

First, PAK200 was not received when expected; therefore, the creation of MULTI was delayed since it was assumed that PAK200 needed only minor revisions to make it compatible to the ASD CYBER system. When PAK200 was received and redesigned, it was found that several programming methods had to be researched to put MULTI into a program compatible with the restrictions placed upon AFIT INTERCOM users.

The use of the FORTRAN V language also presented a barrier which was not eliminated until several months into MULTI's development. This problem concerned the use of CHARACTER statements, first available in FORTRAN V, which produced sporadic errors during MULTI's many modifications. To eliminate the problem, the CHARACTER variable, MULTI, was required to be defined as 10 characters in length to conform to one computer word, rather than be defined as 5 characters in length. This problem was known to FORTRAN V subscribers but was not related to FORTRAN V users.

ASD computer downtime during the summer of 1981, as well as the automatic, unexplained file-purging methods of the ASD CYBER system, delayed the development of a satisfactory program.

The final version of MULTI is the best that can be obtained by control engineer programmers within the time constraints imposed on the development, testing, and simulation of a model, controller and computer package for the newest techniques of the multivariable design of this thesis. MULTI meets all the requirements as outlined at the beginning of the chapter.

Summary

This chapter presents a brief description of the developmental steps used to create MULTI. The discussion begins with the program requirements, emphasis, and constraints. The chapter ends by discussing several developmental problems. The next chapter discusses the MULTI program structure.

V MULTI Program Structure

Introduction

This chapter provides the reader with a brief description of MULTI's internal structure. Since MULTI must be able to operate in $65{,}000_8$ words of core memory, its program structure consists of a group of various modules. Information between these modules is passed by using the data base concept of global storage. To allow MULTI to be fully interactive, the program uses a very simple Program Control Interface. These three aspects of computer programming describe MULTI's program structure.

Overlays

To reduce the amount of memory core required for operation, MULTI has a structure consisting of several modules, called overlays. Each overlay is an executable program and combines with all other overlays to form a complete operational overlay structure. There is one main overlay in MULTI and 13 primary overlays. The main overlay is an Executive Program directing the actual functions of the program to the 13 lower-level primary overlays.

This overlay structure is responsible for MULTI's operation within the defined $65{,}000_8$ memory core limit. The main overlay remains in executable memory at all times. As each different computational or functional requirement of the program is needed, the overlay designed to do the procedure is loaded into executable memory behind the main overlay. When the procedure is finished, the primary overlay is saved and is replaced by the overlay needed for the next procedure.

Data information is passed between overlays in MULTI by the use of labeled common blocks. If a program variable in an overlay is not listed in a labeled common block, its value is not retained when the next overlay is loaded.

A complete description of overlays and how they are used in MULTI can be found in the MULTI Programmer's Manual attached as Appendix B. For more detailed information the reader is referred to the CDC FORTRAN V Reference Manual (Ref 15) and the CDC CYBER Loader Reference Manual (Ref 16).

Data Base

MULTI uses three types of data storage methods which comprise its data base. These three types of storage methods are: local storage, global storage, and mass storage. Mass storage in MULTI entails only the use of sequential-access files.

Local storage is the storage method which is used for program variables that are needed only during the execution of a particular overlay. As mentioned previously, these locally stored values are lost when the overlay is finished executing. MULTI has several matrices which are generated during program execution whose values are destroyed when they are no longer needed.

Global storage is used for program variables whose element values are required to be passed between overlays. All of MULTI's global storage is accomplished using 24 labeled common blocks. The variables defined in each common block are purposely chosen to keep each overlay memory core requirement at a minimum while still allowing for the necessary data transfer between overlay loading.

Sequential-access files are used in MULTI's external data input functions and in MULTI's mass storage capability. A sequential-access, data-input file can be built by any user for use with the program. The specifics and proper formatting of these files is described in detail in the user's manual attached as Appendix A. MULTI automatically creates three separate memory files when the user terminates program operation. A file called "MEMO" is generated to hold all data input values describing the plant of interest; a file called "MEM10" is created to hold all design parameters; a file called "MEM20" is formed to hold simulation parameters. These local files can be stored on the user's permenent file space for future use with MULTI, or can be routed to the system printer after program termination.

Program Control Interface

To explain the interactive features which are incorporated in MULTI, it is necessary to describe the aspects of Program Control Interfacing. MULTI's interface design is similar to the interface found in TOTAL (Ref 17), but is not nearly as complex.

Interfacing Pattern. All MULTI operations are controlled by having the user choose various option numbers which correspond to desired computational functions. Since there is minimal input data verification, the user must adhere strictly to the limitations noted in various parts of the MULTI User's Manual. Every completed option sets an internal "flag" allowing entry into other options. There is a definite pattern for successful program operation which corresponds to a normal design process. The program is designed to require the user to flow from data input, to controller design, to simulation, and finally to testing. Each part of the design can be re-accomplished interactively, but the user is not permitted to proceed in the flow until each preceeding design step is accomplished. However, the user need not test each design.

All options are explained in the MULTI User's Manual.

<u>Data Access and Manipulation</u>. Data access in MULTI is accomplished by in-line program requests and by the use of a special option number range. The user can view certain pertinent matrix combinations and other data of interest by responding affirmatively when the program prints:

```
ENTER "0" TO OBTAIN DATA PRINTOUT
ENTER "1" TO SKIP DATA PRINTOUT...>
```

In some cases, the data cannot be accessed again, since it is contained on a local storage device, unless the user re-enters the same option with the same plant data and design specifications.

The second method of data access is by the use of options having the value of 100 or greater. If data has been entered using OPTION #1, the same data can be accessed for verification by using OPTION #101. This method of data access is incorporated in the final version of the program since earlier versions of the program had only a full, in-line data requesting ability. This original in-line format of displaying data uses too much time during the complete design process. The final data access method allows for easy, selective data access without the complex character verification routines so often used in similar programs.

After data access the user may determine that values are not entered correctly, or for some reason, need to be changed. The user can use one of two means to change the data. The option related to the data in question can be re-entered or the user can terminate MULTI and edit the program-created data files. This last means of changing data is of special interest when changing element values of the plant's $\underline{A}$, $\underline{B}$, $\underline{C}$, and $\underline{D}$ matrices.

Error Detection and Recovery

The error detection and recovery aspects of MULTI are somewhat limited. Since the main emphasis of the thesis is not perceived to entail the complete design of a computer simulation package, MULTI's design is not based upon the ability to protect the user from input errors. MULTI is designed to alert a user, and recover, when program flow is not in the proper order, or when certain plant deficiencies require the use of a certain controller design.

Input errors such as character entries for numerical data, and vice versa, are all detected by the CYBER error detection capabilities. In these cases the user is allowed to re-enter the proper information and continue.

MULTI restricts the user to a system of no more than ten states, ten inputs, and ten outputs and 2nd order actuators and sensors.

Simultaneous simulations can be run for up to two different sampling times while plots can be generated which display up to four superimposed curves. When the user's requests exceed these limitations, the entire MULTI program is aborted. It is suggested that any MULTI user become completely familiar with the user's guide for the program.

When MULTI is aborted the CYBER system relates the nature of the error which causes the abnormal termination of the program.

Summary

This chapter summarizes the MULTI program structure. MULTI's main feature is its overlay structure. As each specific option is requested, the overlay responsible for the required calculations is loaded into operational memory core. Error detection and recovery capabilities are shared between the MULTI program and the CYBER computer system.

The following chapter contains a control law design for the Mach 0.18 flight condition of the A-7D Digitac II aircraft. This control law is designed and simulated using MULTI.

VI Control Law Design

Introduction

This chapter summarizes the trials which are made to obtain a multivariable, digital control law for the 0.18 Mach flight condition of the A-7D Digitac II aircraft. The discussion begins with a brief look at two early aircraft models which cannot be used. The aircraft equations for the final 0.18 Mach model are presented next, and the steps used to synthesize the final control law are outlined. The chapter ends by showing what response effects occur when various control law parameters are changed.

Preliminary Models

Several state and output equation forms are derived as various models are developed. When an unforseen obstacle prevents a suitable design for one set of equations, the equations are restructured to bypass the obstacle. This section discusses two of the preliminary models which are tried.

The reader is referred to the Paschall thesis (Ref 7) for information on other state and output equation forms.

Unknown Plant. As noted in the section on unknown plants in Chapter III, if the state equations are known, the unknown plant method can be used to develop a controller after obtaining the $\underline{G}(0)$ matrix. The computer program, MULTI, forms this matrix automatically after the

$\underline{A}$, $\underline{B}$, $\underline{C}$, and $\underline{D}$ plant matrices are entered. For the original aircraft model with split control surfaces, eight states, eight inputs, and eight outputs (Ref 7:Appendix A), the $\underline{G}(0)$ matrix is:

$$\underline{G}(0) = \left[\begin{array}{rrrrrrrr}
.1709E{+}04 & .1709E{+}04 & 0. & -.5164E{+}03 & .5164E{+}03 & .5663E{+}02 & .5663E{+}02 & .6232E{+}03 \\
.1789E{+}02 & .1789E{+}02 & 0. & -.5414E{+}01 & .5414E{+}01 & .5448E{+}00 & .5448E{+}00 & .6003E{+}01 \\
-.1389E{-}14 & -.1389E{-}14 & 0. & .4628E{-}15 & -.4628E{-}15 & -.5231E{-}16 & -.5231E{-}16 & -.6384E{-}15 \\
.1668E{+}02 & .1668E{+}02 & 0. & -.5115E{+}01 & .5115E{+}01 & .5180E{+}00 & .5180E{+}00 & .5808E{+}01 \\
.2359E{-}01 & -.2359E{-}01 & .2813E{-}01 & -.1014E{+}00 & .1014E{+}00 & -.1078E{-}01 & .1078E{-}01 & .2644E{-}01 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
.2708E{-}02 & -.2708E{-}02 & -.1532E{+}00 & -.2402E{-}01 & .2402E{-}01 & -.4882E{-}02 & .4882E{-}02 & .4208E{-}02 \\
.5493E{-}01 & -.5493E{-}01 & -.3567E{+}01 & -.5643E{+}00 & .5643E{+}00 & -.1153E{+}00 & .1153E{+}00 & .8210E{-}01
\end{array}\right] \qquad (30)$$

Although the system has an equal number of inputs and outputs, the $\underline{G}(0)$ matrix does not have rank equal to the number of system outputs. Thus, the model does not meet all the criteria for control law design by the unknown plant method.

Augmented State Equations

A major criterion for the regular plant design method is that the first Markov parameter has full rank. In the case of the A-7D

Digitac II model, using the split control surface configuration, it is found that this requirement is not met. Although the system states are chosen so that the $\underline{C}_2$ matrix has full rank, the rank of the $\underline{B}_2$ matrix is always rank deficient due to the elements associated with the aircraft's split flight control surfaces.

It is initially proposed that if the system is augmented with actuator transfer functions, the $\underline{B}_2$ matrix can be adjusted so that full rank can be produced. When this is accomplished, a full rank $[\underline{C}_2\ \underline{B}_2]$ matrix results. However, it is found that for the states chosen for the system, the model contains transmission zeros which lie on the z-plane unit circle. This is noted in early design trials which produce unstable states for the sampling frequency of interest.

Final Model

The final model contains eight states, six inputs, and six outputs. Except for the horizontal stabilator, the flight control surfaces are recombined. The output equations use flight path angle as an output where:

$$\gamma = (\theta - \alpha) \tag{31}$$

but do not directly include angle of attack, α . Roll rate, p, and pitch rate, q, are not used as outputs. The final state and output equations for the Mach 0.18 flight condition are:

$$\begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \dot{u} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\beta} \\ \dot{p} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0. & 0. & 0. & 0. & .1000E{+}01 & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & .1000E{+}01 & 0. \\ -.3220E{+}02 & 0. & -.6800E{-}02 & -.4130E{+}00 & 0. & 0. & 0. & 0. \\ 0. & 0. & -.2134E{-}03 & -.4189E{+}00 & .1000E{+}01 & 0. & 0. & 0. \\ 0. & 0. & .8290E{-}03 & -.6890E{-}01 & -.1655E{+}02 & 0. & 0. & 0. \\ 0. & .1630E{+}00 & 0. & 0. & 0. & -.1528E{+}00 & .1055E{+}00 & -.7291E{+}00 \\ 0. & 0. & 0. & 0. & 0. & -.7143E{+}01 & -.1980E{+}01 & .8104E{+}00 \\ 0. & 0. & 0. & 0. & 0. & .1561E{+}01 & -.4254E{+}01 & -.1800E{+}02 \end{bmatrix} \begin{bmatrix} \theta \\ \phi \\ u \\ \alpha \\ q \\ \beta \\ p \\ r \end{bmatrix}$$

$$
+\begin{bmatrix}
0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. \\
0. & -.1429E{+}02 & -.1429E{+}02 & 0. & -.2322E{+}01 & -.4926E{+}01 \\
0. & -.3280E{+}00 & -.3280E{+}00 & 0. & .1630E{+}00 & -.9573E{+}00 \\
0. & -.4774E{+}01 & -.4774E{+}01 & 0. & -.3504E{+}00 & -.1186E{+}00 \\
.2684E{+}00 & 0. & 0. & -.1527E{+}00 & 0. & 0. \\
.2379E{+}01 & .1080E{+}01 & -.1080E{+}01 & .1387E{+}02 & 0. & .1558E{+}01 \\
-.9166E{+}01 & -.2440E{-}01 & .2440E{-}01 & .3142E{+}01 & 0. & .2840E{-}01
\end{bmatrix}
\begin{bmatrix}
\delta_r \\
\delta_{H_{right}} \\
\delta_{H_{left}} \\
\delta_a \\
\delta_s \\
\delta_f
\end{bmatrix}
\qquad (32)
$$

$$\begin{bmatrix} \gamma \\ u \\ \theta \\ \beta \\ r \\ \phi \end{bmatrix} = \begin{bmatrix} .1000E{+}01 & 0. & 0. & -.1000E{+}01 & 0. & 0. & 0. & 0. \\ 0. & 0. & .1000E{+}01 & 0. & 0. & 0. & 0. & 0. \\ .1000E{+}01 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & .1000E{+}01 & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & .1000E{+}01 \\ 0. & .1000E{+}01 & 0. & 0. & 0. & 0. & 0. & 0. \end{bmatrix} \begin{bmatrix} \theta \\ \phi \\ u \\ \alpha \\ q \\ \beta \\ p \\ r \end{bmatrix} \tag{33}$$

Control Law Design

The control law synthesis is accomplished using the interactive computer program, MULTI, found in Appendix A. The following text is a condensed narrative of the process used to find an initial control law for the aircraft model described by Eqs. (32) and (33). The design does not include input control limitations, sensors, or actuators.

As an initial trial, the aircraft equations are entered into the program and all design parameters are set to unity values. The responses for a flight path angle command are found to be oscillitory and unstable as shown in Fig. 2.

The parameter, α, representing the ratio of integral of error feedback to direct error feedback, is changed until the response for the flight path angle command successfully follows the command input. The output response, with $\alpha = 0.10$, is shown in Fig. 3, and shows an overshoot of approximately 35%.

With $\alpha = 0.1$, variations in the ϵ parameter are considered next. Trials are made with ϵ varying from 0.01 to 2.0. It is found that a trade-off in peak values and settling time occurs for different combinations of the α and ϵ values. As a result, the terminal plot of Fig. 4 is obtained with $\alpha = 1.0$ and $\epsilon = 0.05$.

At this point, since tracking is shown for a commanded flight path angle, all other inputs are commanded and the responses noted. As a guideline, it is decided that sufficient tracking can be attained if the response shows less than 25% overshoot with a settling time of five seconds for all input commands. Except for the settling time in the yaw rate state, the specifications are met satisfactorily. However, some interaction is noted in the other states.

The diagonal weighting matrix elements, σ_i , are then adjusted in an attempt to eliminate the interaction between states. For all the simulation trials with various combinations and values for the weighting matrix elements, it is noted that the interaction cannot be significantly reduced for one commanded input without degrading the basic tracking qualities when another input is commanded.

Thus, the "optimal" control law found from all the simulation trials is defined by the following parameters:

$$T = 0.01 \tag{34}$$

$$\alpha = 1.0 \tag{35}$$

$$\Sigma = \text{diag}\,[\;1\;,\;1\;,\;1\;,\;1\;,\;1\;,\;1\;] \tag{36}$$

$$\epsilon = 0.05 \tag{37}$$

which give:

$$\underline{K}_0 = \underline{K}_1 = \left[\begin{matrix} .6462E{-}03 & -.9494E{-}04 & .1314E{-}02 \\ -.2659E{-}01 & .5347E{-}02 & -.9227E{-}01 \\ .3395E{-}01 & -.3547E{-}02 & .3084E{-}01 \\ .1136E{-}02 & -.1669E{-}03 & .2310E{-}02 \\ -.1107E{+}00 & -.2300E{-}01 & .2448E{+}00 \\ .3085E{-}01 & -.4532E{-}02 & .6273E{-}01 \end{matrix}\right.$$

$$\left.\begin{matrix} -.3727E{+}00 & -.1646E{-}01 & -.1488E{-}02 \\ .6723E{+}01 & .2041E{+}00 & .1111E{+}00 \\ -.6723E{+}01 & -.2041E{+}00 & -.1111E{+}00 \\ -.9826E{+}00 & -.2894E{-}01 & -.2616E{-}02 \\ .1493E{-}12 & .4532E{-}14 & .2467E{-}14 \\ .4201E{-}13 & .1275E{-}14 & .6941E{-}15 \end{matrix}\right] \tag{38}$$

The measurement matrix for the irregular plant design is:

$$\underline{M} = \begin{bmatrix} 0. & 0. \\ 0. & 0. \\ 0.25 & 0. \\ 0. & 0. \\ 0. & 0. \\ 0. & 0.25 \end{bmatrix} \tag{39}$$

Figures 5 thru 10 show the system responses for each commanded input. The results a summarized in Table II. The forward velocity interaction during a pitch angle command is not included in Fig. 7.

Robustness. The "optimal" control law shows that adequate tracking for the Mach 0.18 can be obtained for each commanded input. The control law is then used to test the responses for flight path angle, forward velocity, and roll angle commands with the Mach 0.6 and 0.8 models as shown in Figs. 11, 12, and 13. The three responses reveal that only forward velocity tracking is achieved over the entire range of Mach numbers.

Failed Control Surfaces. A test of state responses is made with the right horizontal stabilator disabled. To accomplish this condition, the elements of the second column of the B matrix (Eq. 32) are changed to zero values. Since this surface failure may result in both longitudinal and lateral instability, two tests are made. The first trial includes a flight path angle change; the result is shown in Fig. 14. The second trial is made using a roll angle command as the input, and the response is shown in Fig. 15. Both figures

TABLE II
Summary of Controller Design Results

COMMAND	APPROXIMATE OVERSHOOT	APPROXIMATE SETTLING TIME	INTERACTION
Flight Path Angle	6%	1.75 sec	None
Forward Velocity	11%	2.25 sec	None
Pitch Angle	17%	4.50 sec	Forward Velocity
Sideslip Angle	18%	2.75 sec	Yaw Rate Roll Angle
Yaw Rate	0% (over damped)	10.0 sec	None
Roll Angle	8%	1.0 sec	Sideslip Angle Yaw Rate

demonstrate that the control law does not provide stability after a right stabilator surface failure. Although Fig. 14 and Fig. 15 may be interpreted as showing stability until three seconds have elapsed, the magnitude of the independent axis values must be considered. Figure 16 shows that the response to a roll angle command is unstable at two seconds.

Control Surface Inputs. An analysis of the control law design is not complete without mentioning the control surface deflections that are produced for the various commands. As mentioned in the Control Law Design section, the simulations for this chapter are run without control surface limitations. The MULTI computer program allows the user to introduce control limits and shows the number of times that these limitations are exceeded during the simulation run. However, if the control surfaces are actually limited and larger control surface deflections are dictated by the control law, the system may become unstable. Thus, the control limit aspect of MULTI should be used only to provide an indication that a surface has tried to exceed a limitation, and not be interpreted as a method to help the control law eliminate large surface deflections.

For all the simulated control law designs, all flight control surfaces are found to stay within an allowable deflection range. There is one exception, that being the left and right horizontal stabilator deflections produced during a yaw rate command. These control surface values exceed 60° deflections during all simulations. In fact, control law parameter variations, which produce high-tracking qualities with no overshoot, require stabilator deflections as high as 180°. Left and right surface deflections for the horizontal stabilator are also noted to be non-symmetric in some cases (see Fig. 17).

Parameter Variations

This section gives a brief discussion of the effects of changing (1) the α control law parameter, (2) the ϵ control law parameter, (3) the sampling period, T, and (4) the measurement matrix values. This discussion is applicable only to the Mach 0.18 flight condition models presented in this thesis, and is not necessarily valid for other system models. The effects of varying the weighting matrix diagonal elements is not presented since (in the case of this aircraft model) element changes do not produce response changes that are beneficial to all input commands and, thus, no general trends can be shown. It also must be pointed out that each time a control law parameter variation is made, a new control law must be generated and the responses of all inputs and all outputs should be investigated for all input commands.

The effect of changing the α parameter is shown previously in Figs. 2 and 3. In addition to the response changes that were previously discussed, Fig. 18 shows that if $\alpha = 0.01$, the response for a flight path angle command is unstable.

Figure 19 shows a typical response change, for a variation in the ϵ parameter, when yaw rate is commanded. As is the case with the yaw rate response, the other response plots show a significant decrease in rise time.

Figure 20 demonstrates the responses noted for two different sampling times. As expected, the faster sampling frequency produces a faster response in all cases.

For the Mach 0.18 flight condition, the response for a flight path angle command is investigated for three different values of measurement matrix elements (0.1, 0.25, 0.5). It is found that all three responses are identical for variations in the measurement matrix.

Summary

This chapter presents three of the aircraft models that are used in this thesis. Since it is impossible to discuss all the simulations that are made, only the development of the final or "optimal" control law for the irregular plant model at Mach 0.18 is presented in detail.

The final chapter summarizes the results and conclusions of the entire thesis effort. It also provides some suggestions for further study of multivariable, digital control law design which might be accomplished as an extension to the thesis study.

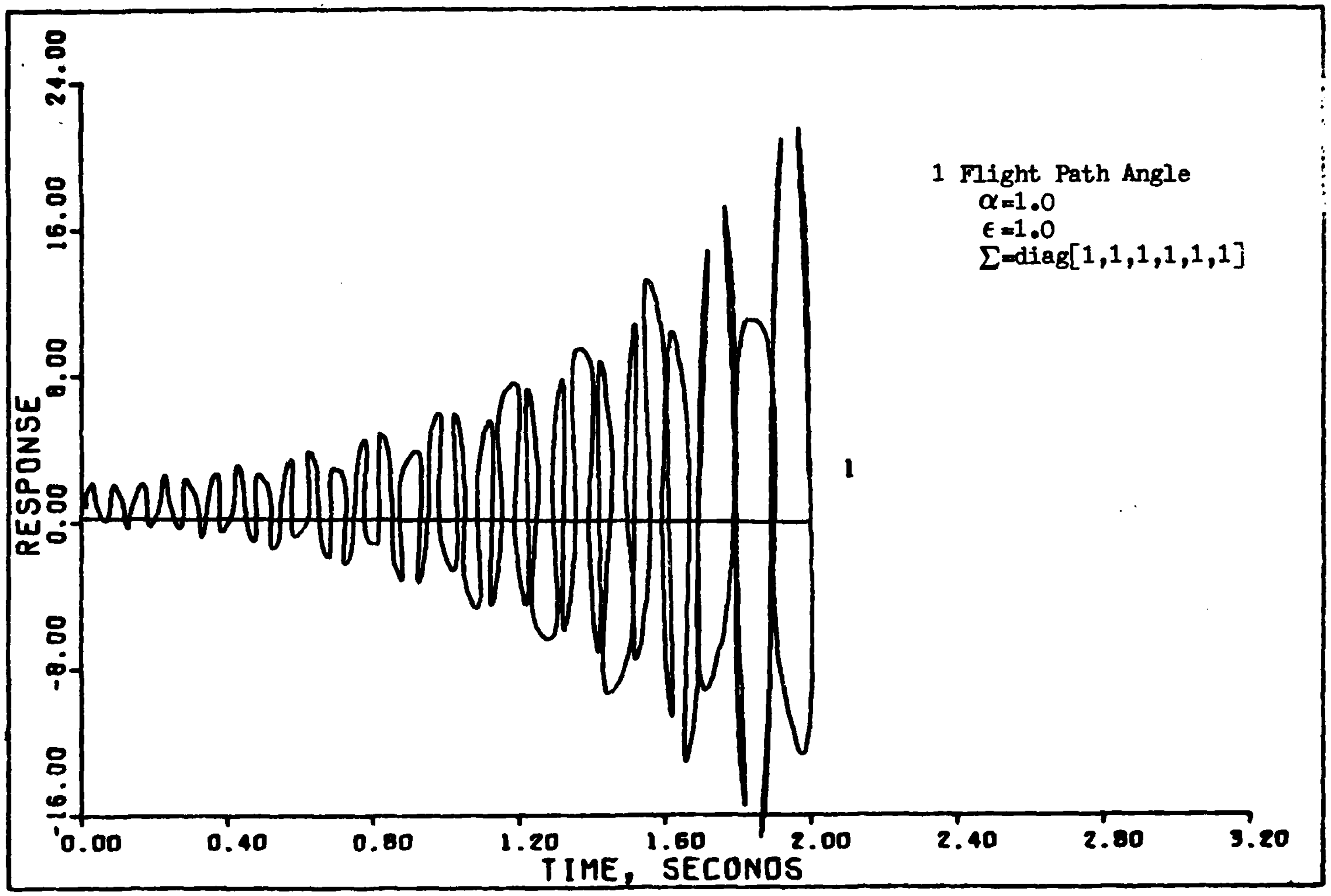


FIG. 2 MACH 0.18 FLIGHT CONDITION, FLIGHT PATH ANGLE COMMAND

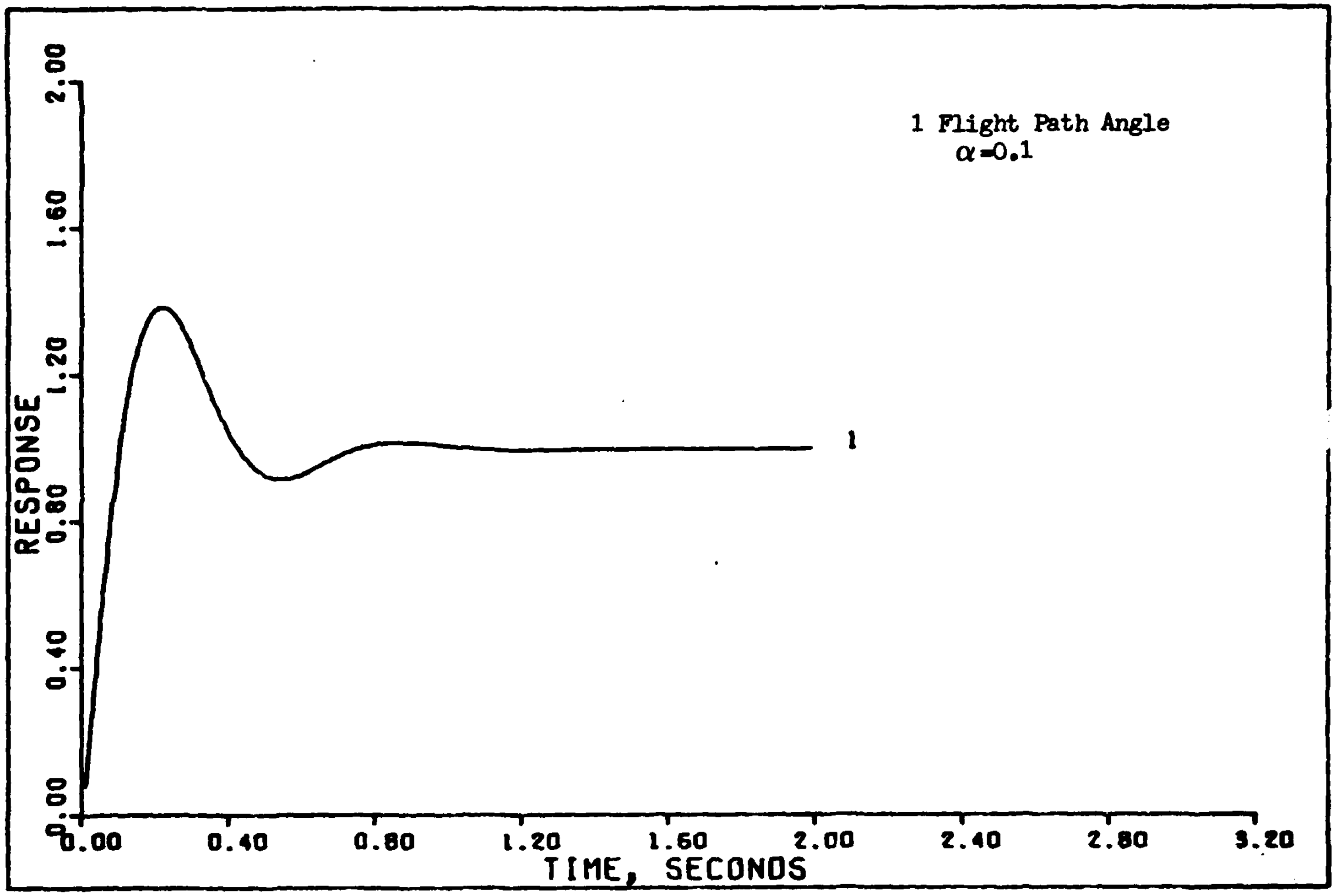


FIG. 3 MACH 0.18 FLIGHT CONDITION, FLIGHT PATH ANGLE COMMAND

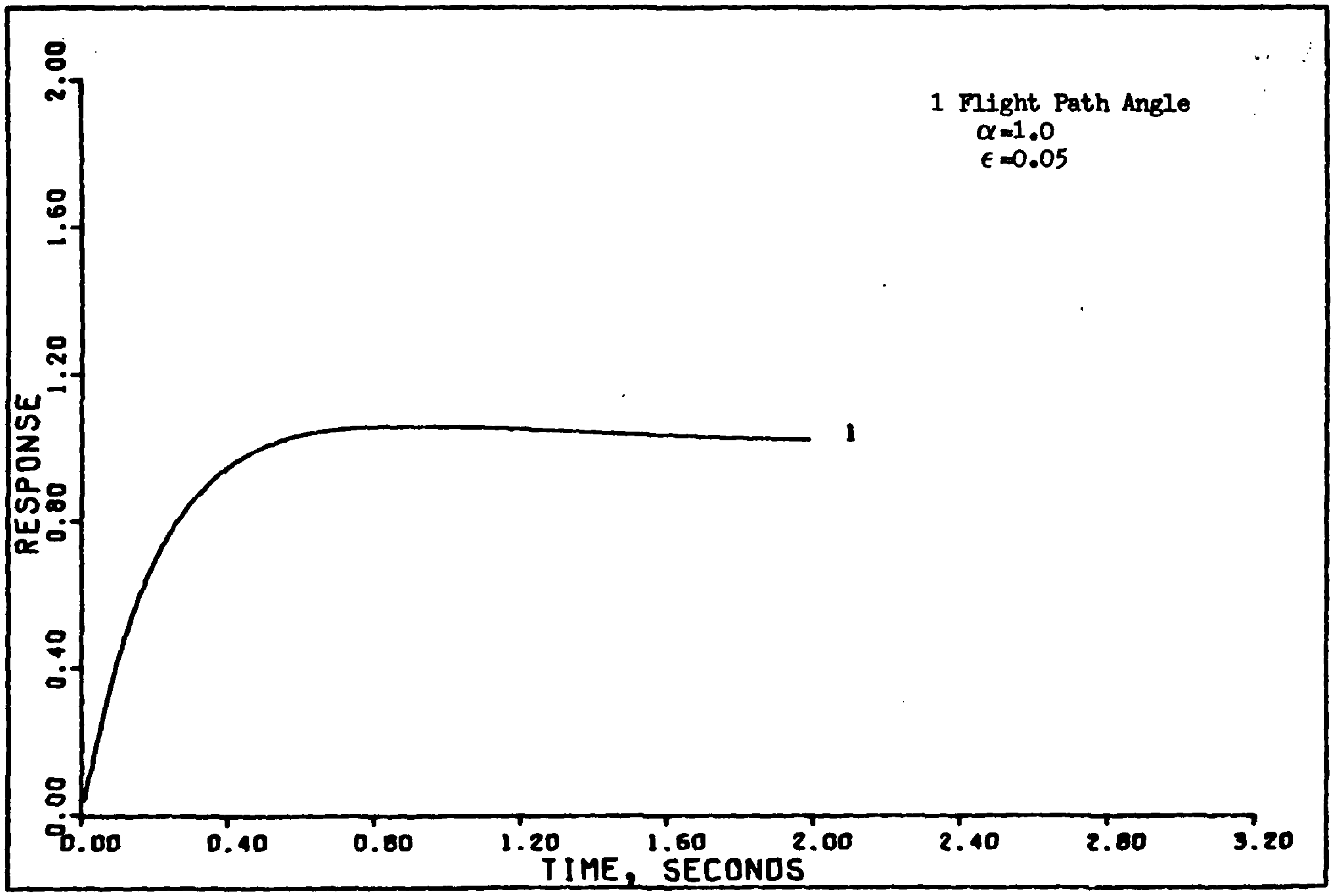


FIG. 4 MACH 0.18 FLIGHT CONDITION, FLIGHT PATH ANGLE COMMAND

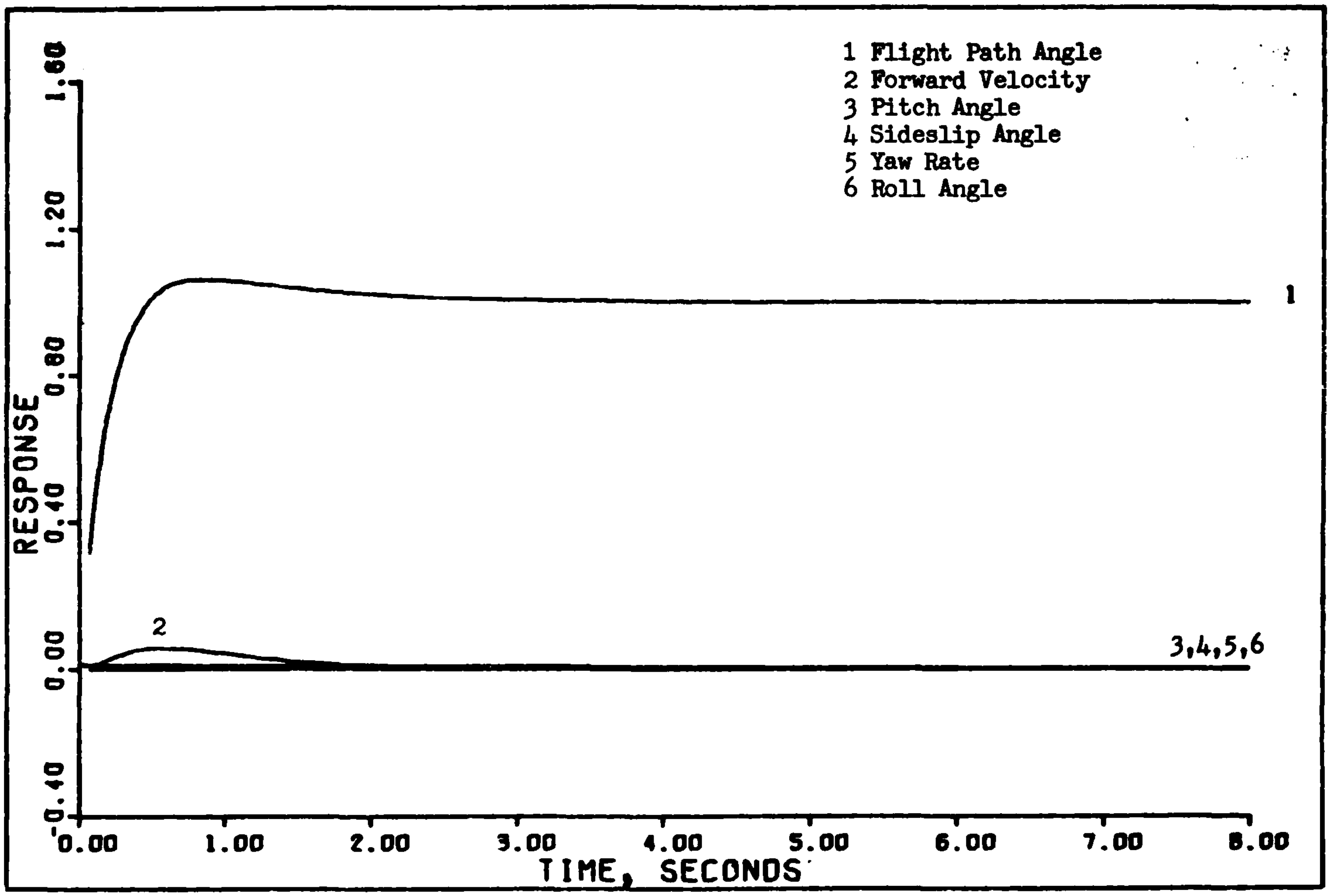


FIG. 5 MACH 0.18 FLIGHT CONDITION, FLIGHT PATH ANGLE COMMAND

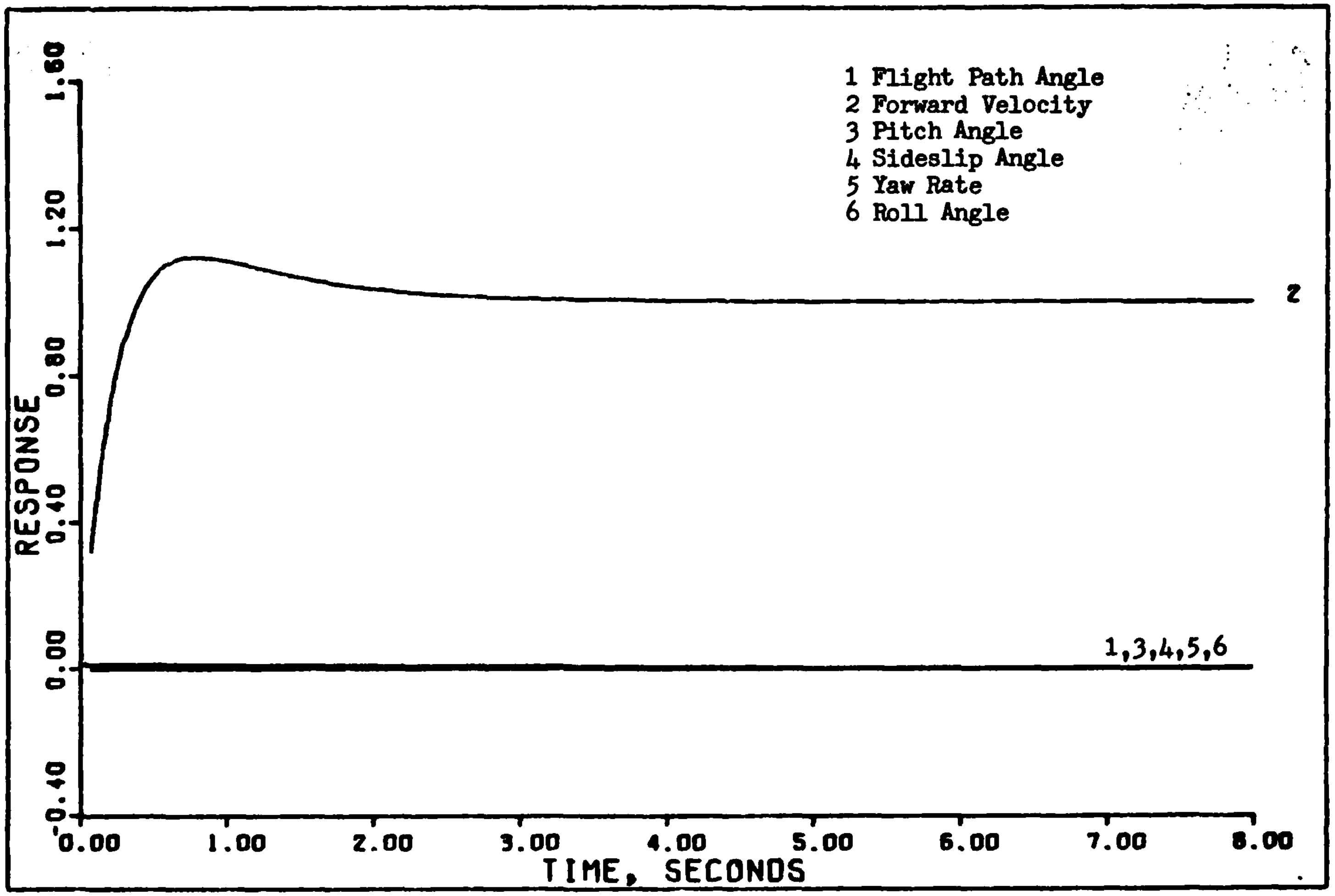


FIG. 6 MACH 0.18 FLIGHT CONDITION, FORWARD VELOCITY COMMAND

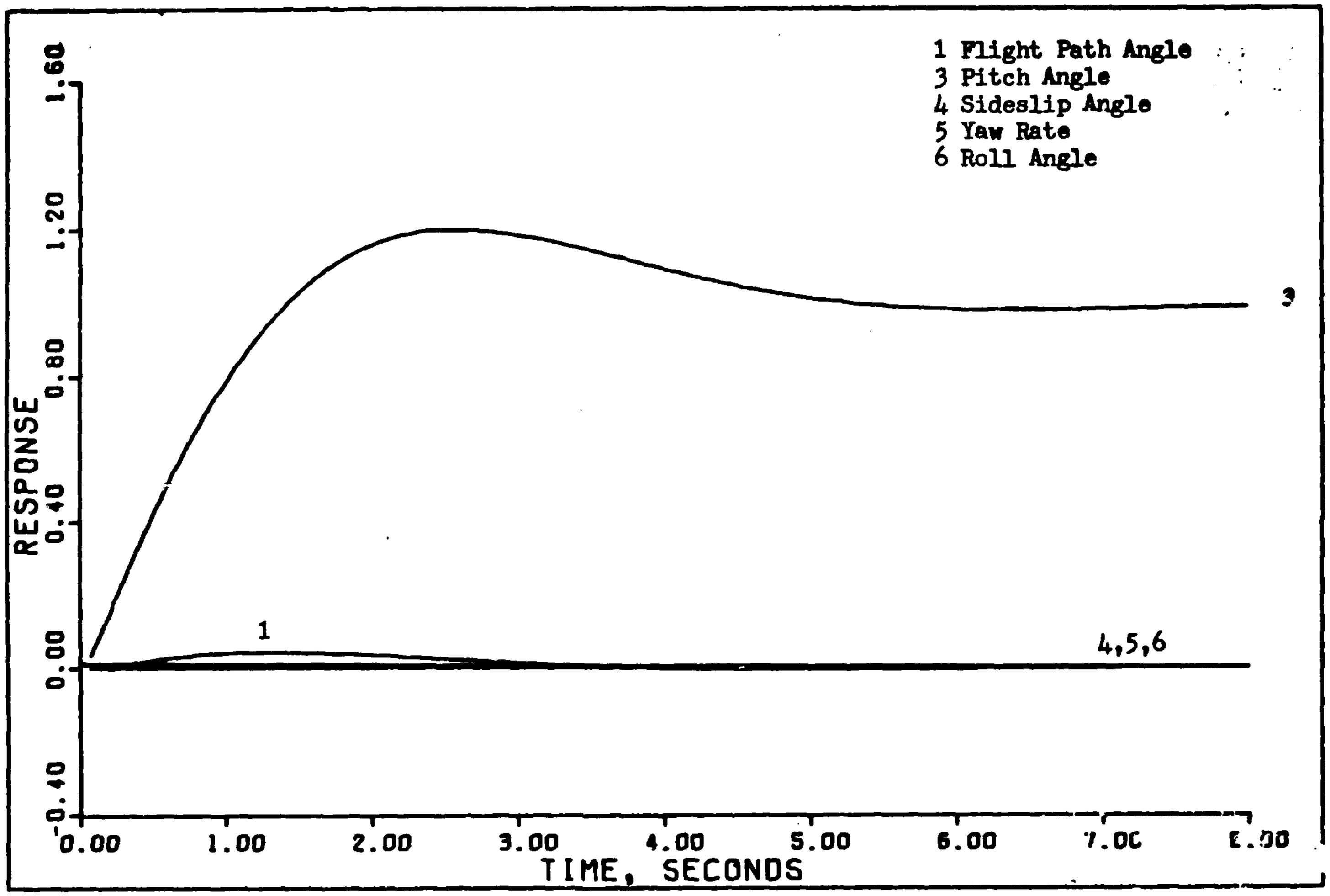


FIG. 7 MACH 0.18 FLIGHT CONDITION, PITCH ANGLE COMMAND

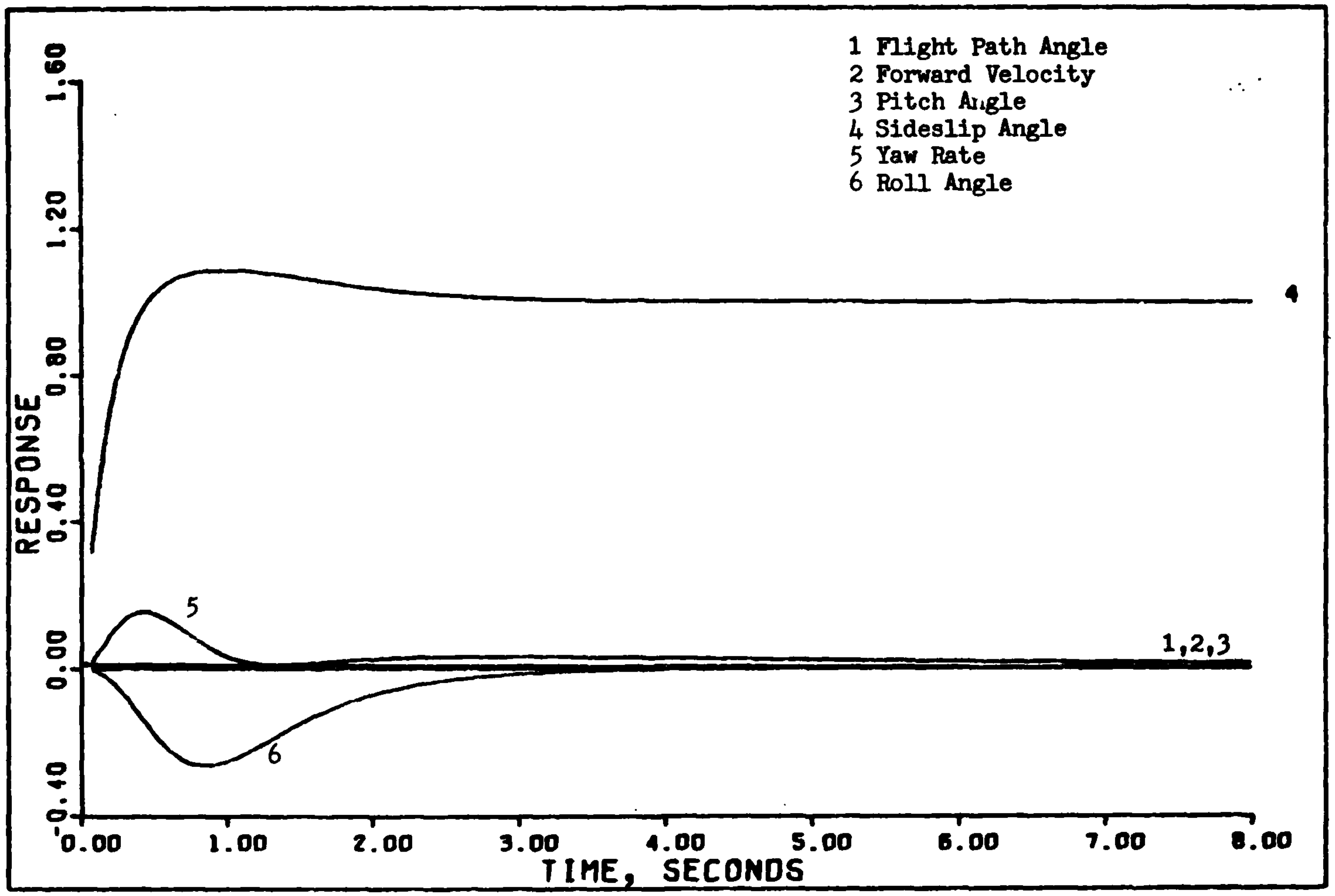


FIG. 8 MACH 0.18 FLIGHT CONDITION, SIDESLIP ANGLE COMMAND

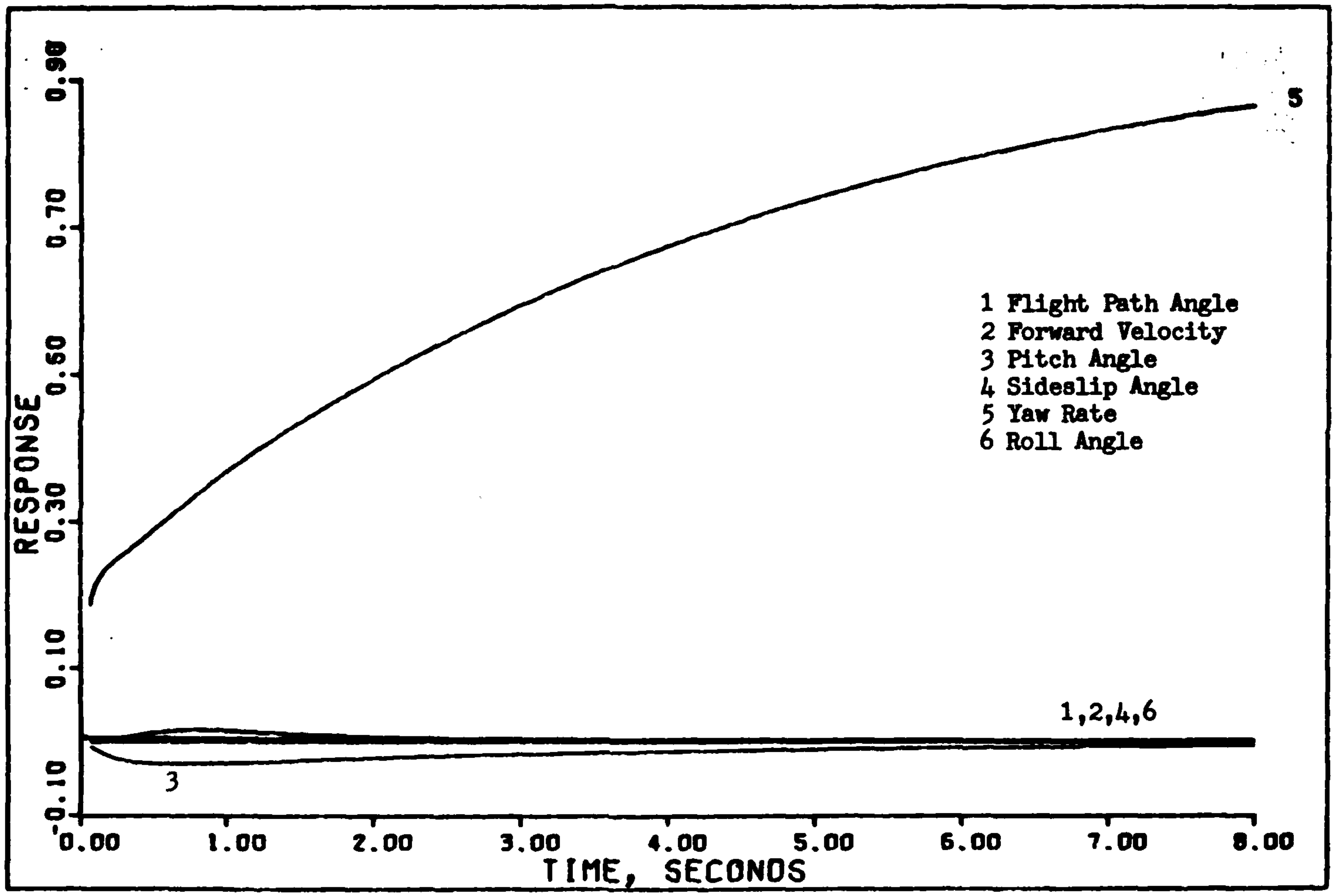


FIG. 9 MACH 0.18 FLIGHT CONDITION, YAW RATE COMMAND

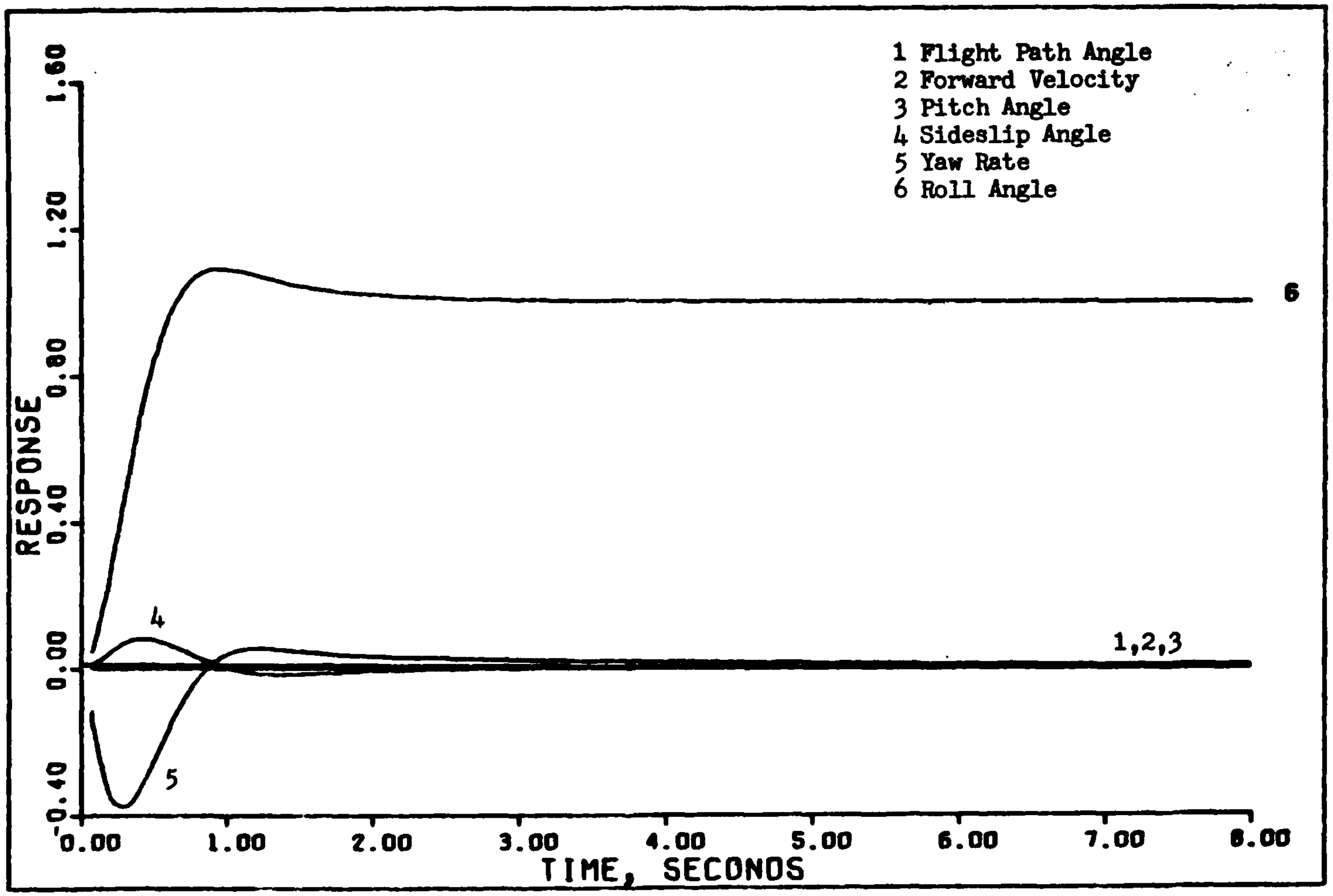


FIG. 10 MACH 0.18 FLIGHT CONDITION, ROLL ANGLE COMMAND

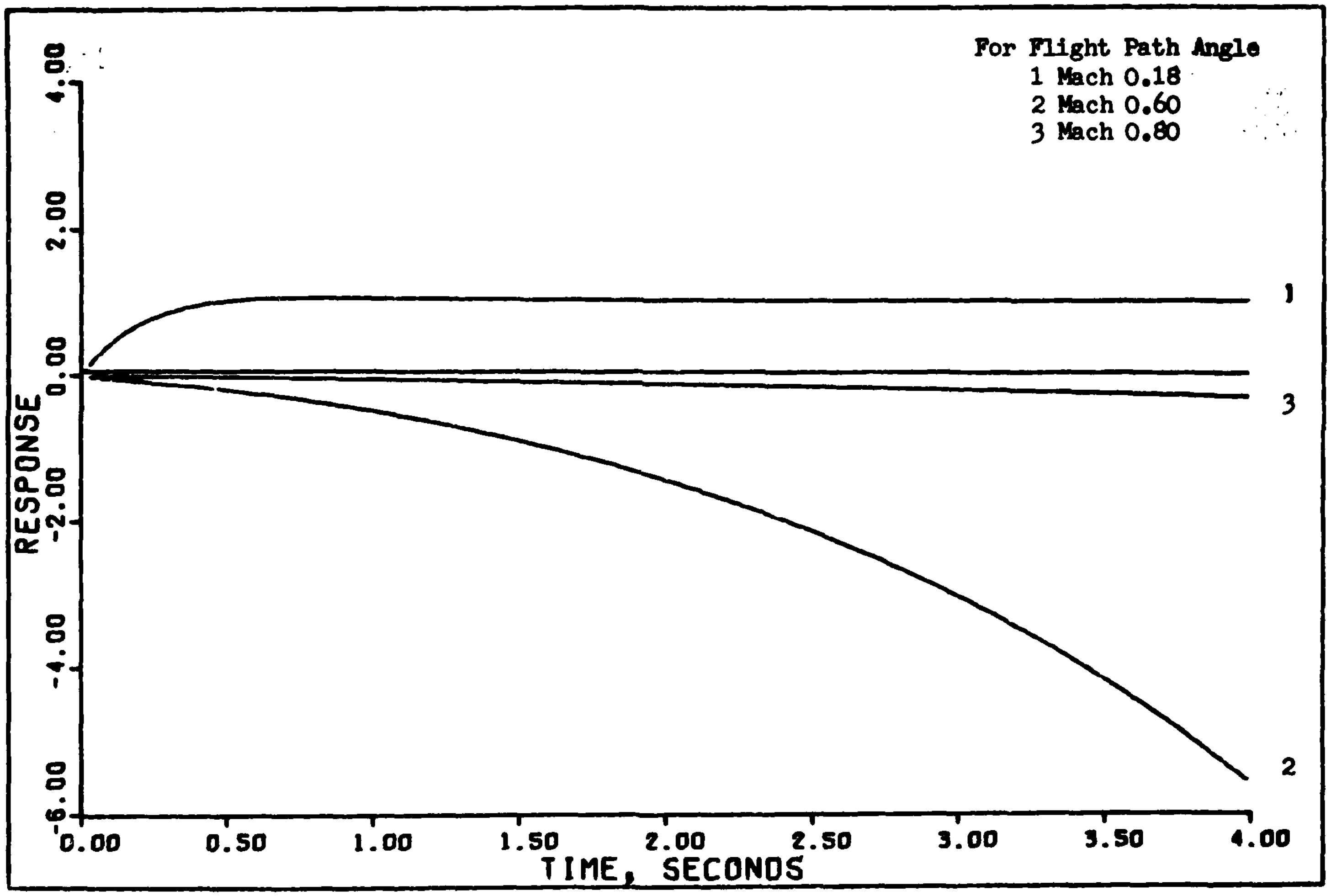


FIG. 11 MULTIPLE FLIGHT CONDITIONS, FLIGHT PATH ANGLE COMMAND

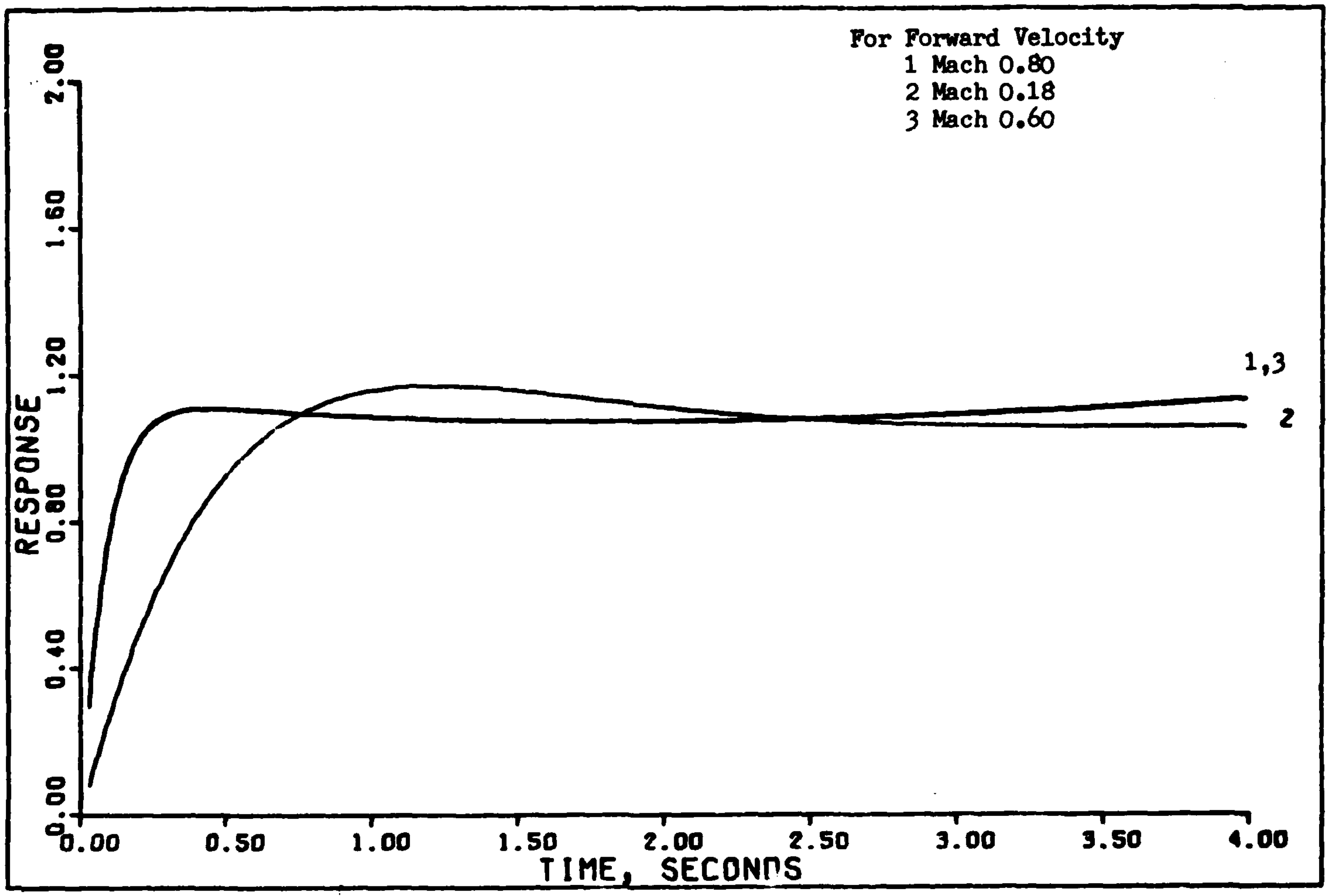


FIG. 12 MULTIPLE FLIGHT CONDITIONS, FORWARD VELOCITY COMMAND

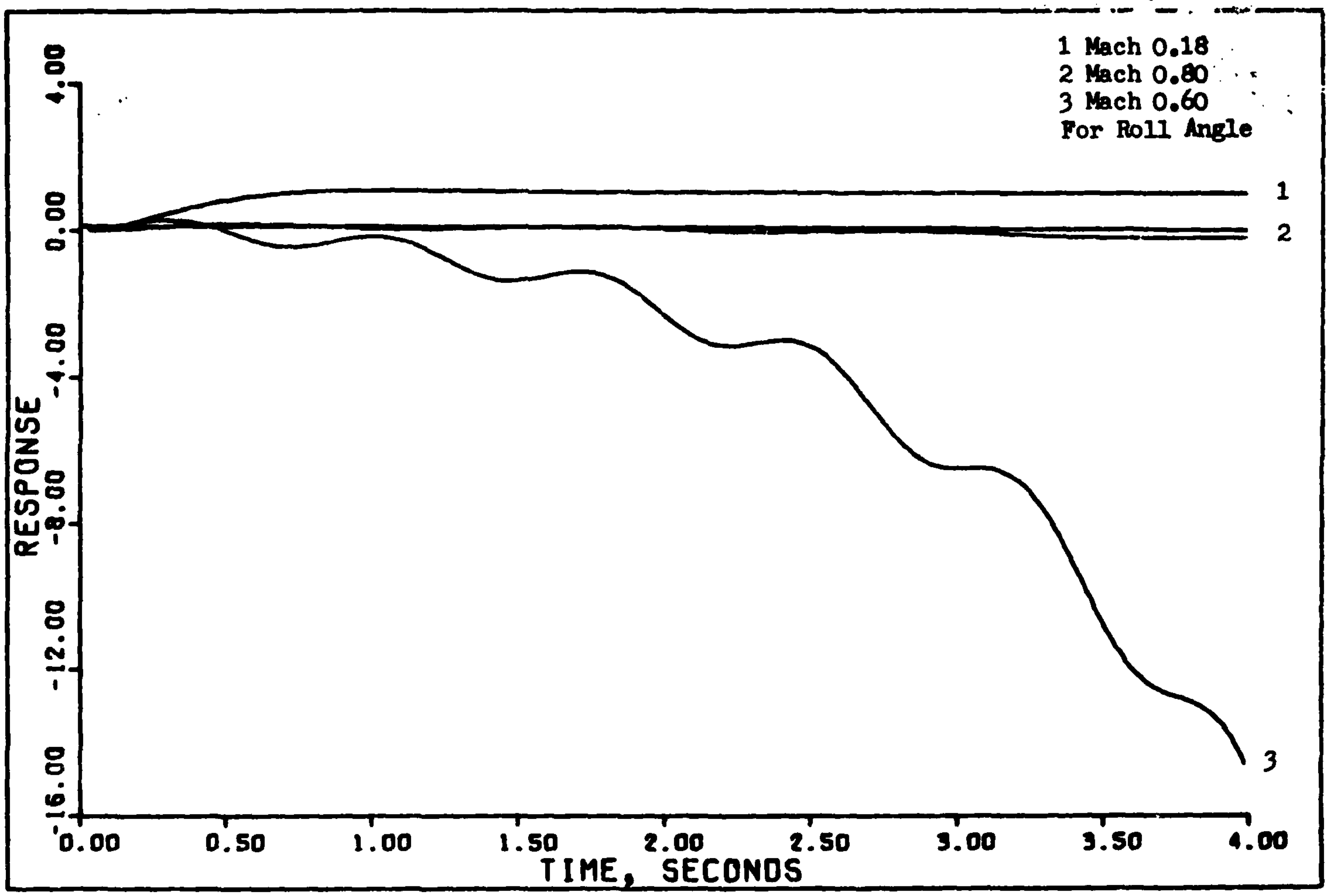


FIG. 13 MULTIPLE FLIGHT CONDITIONS, ROLL ANGLE COMMAND

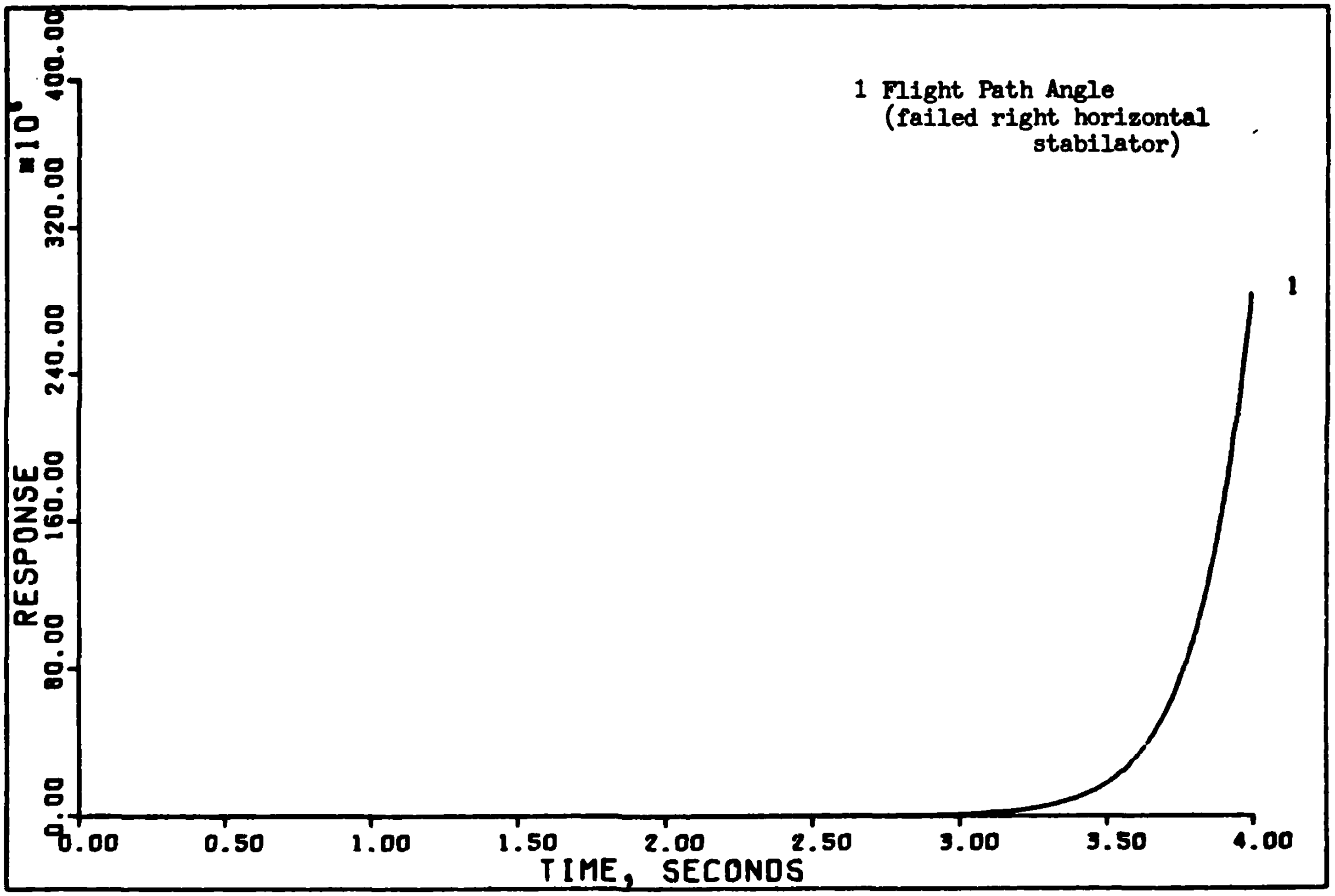


FIG. 14 MACH 0.18 FLIGHT CONDITION, FLIGHT PATH ANGLE COMMAND

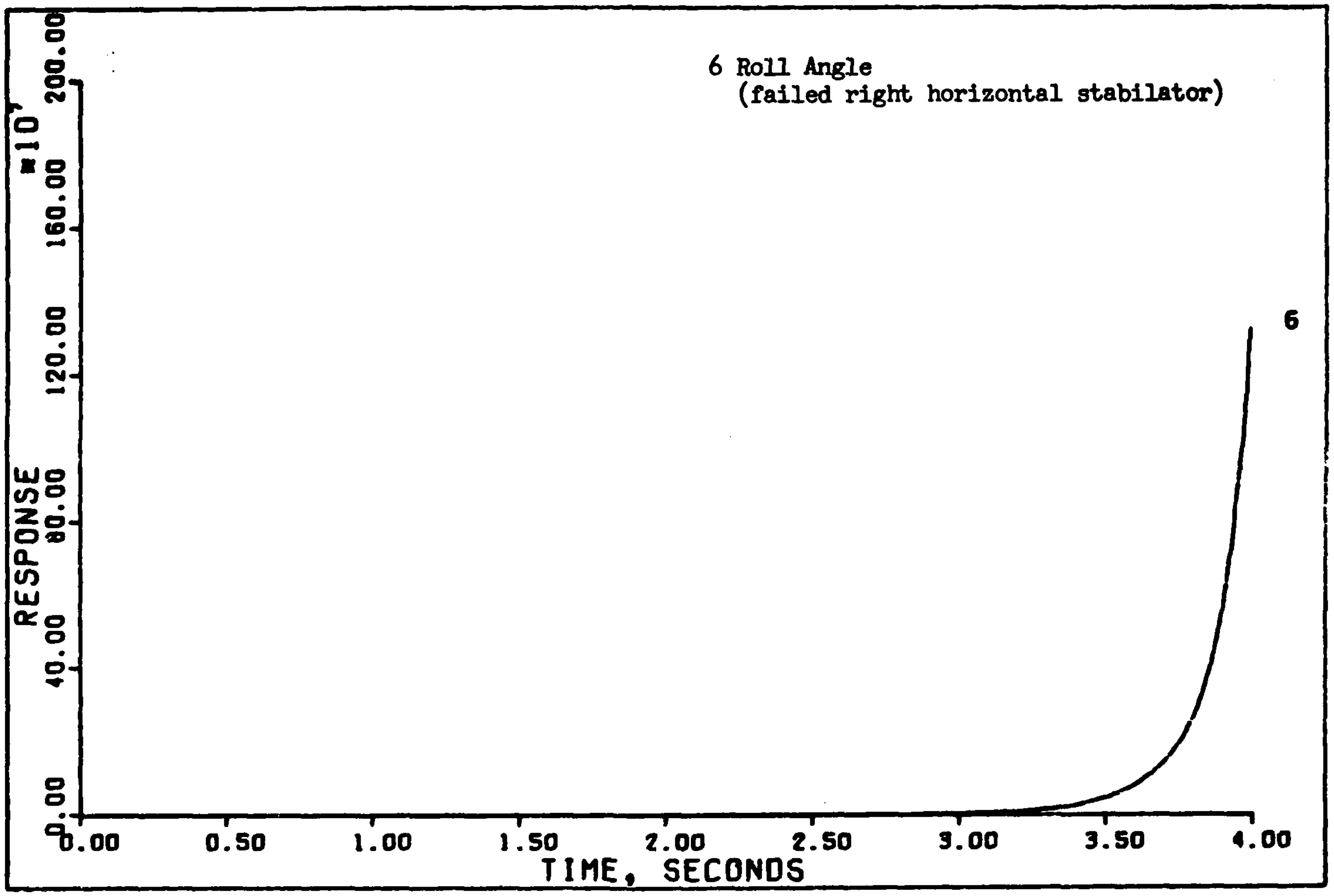


FIG. 15 MACH 0.18 FLIGHT CONDITION, ROLL ANGLE COMMAND

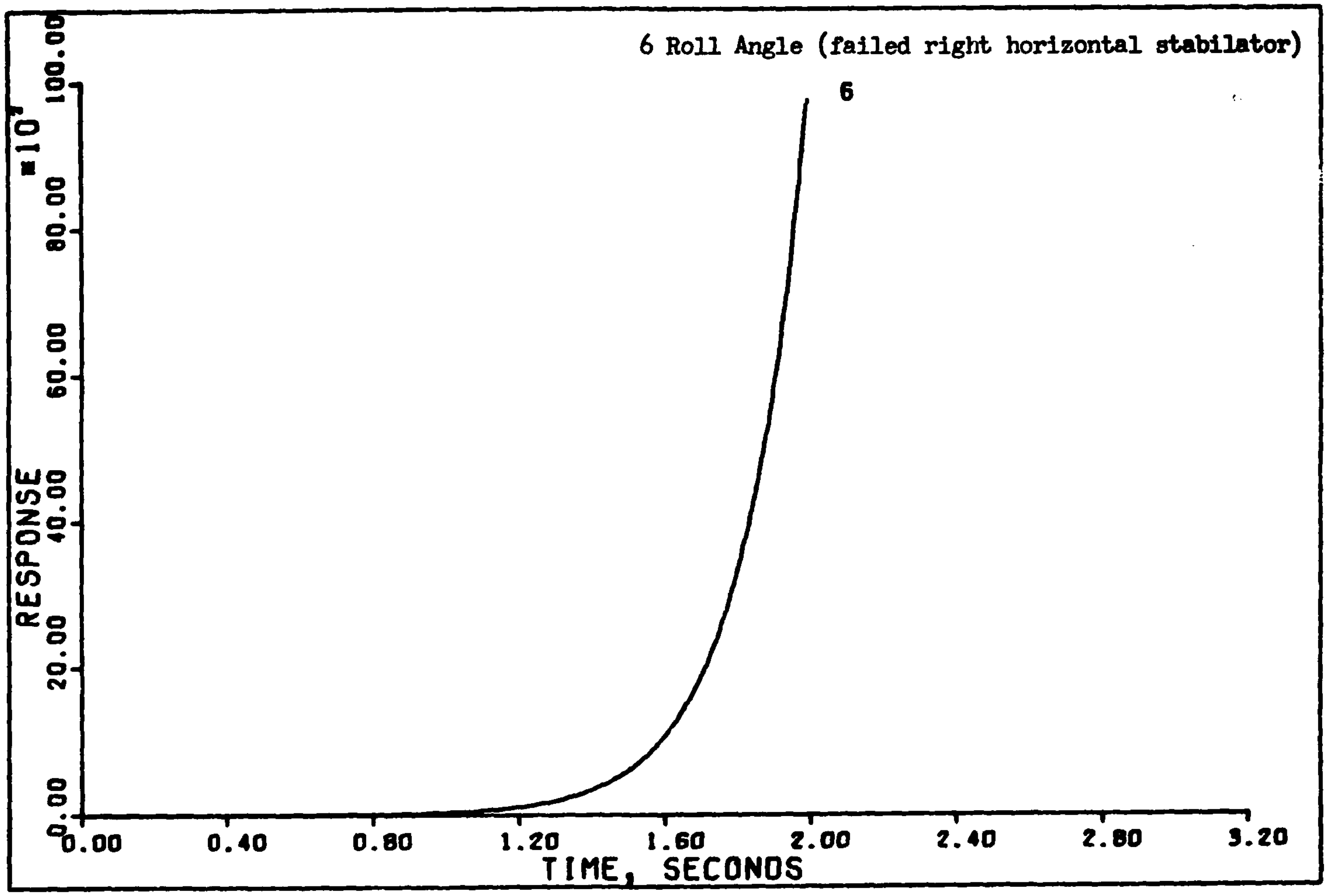


FIG 16. MACH 0.18 FLIGHT CONDITION, ROLL ANGLE COMMAND

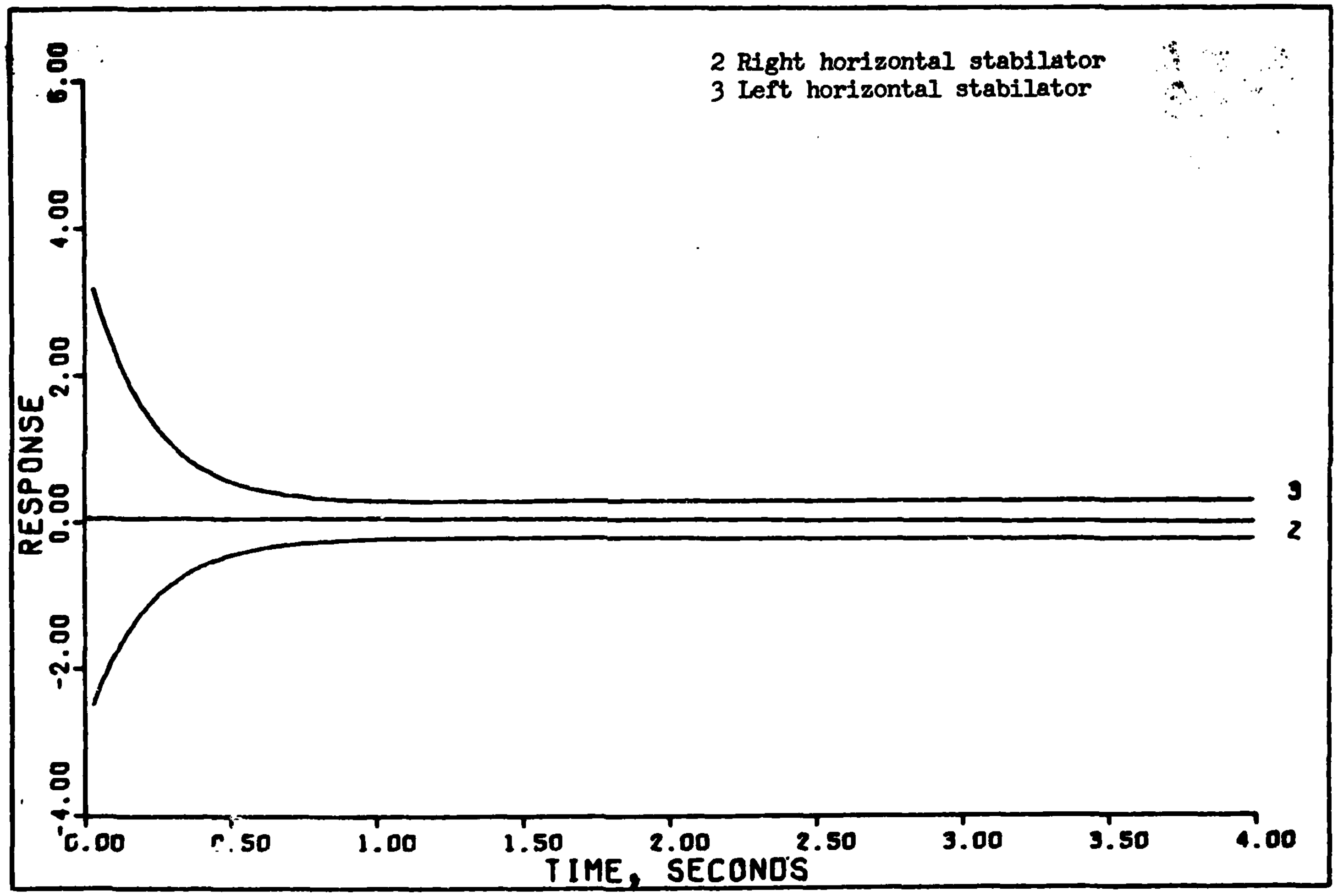


FIG. 17 MACH 0.18 FLIGHT CONDITION, FLIGHT PATH ANGLE COMMAND

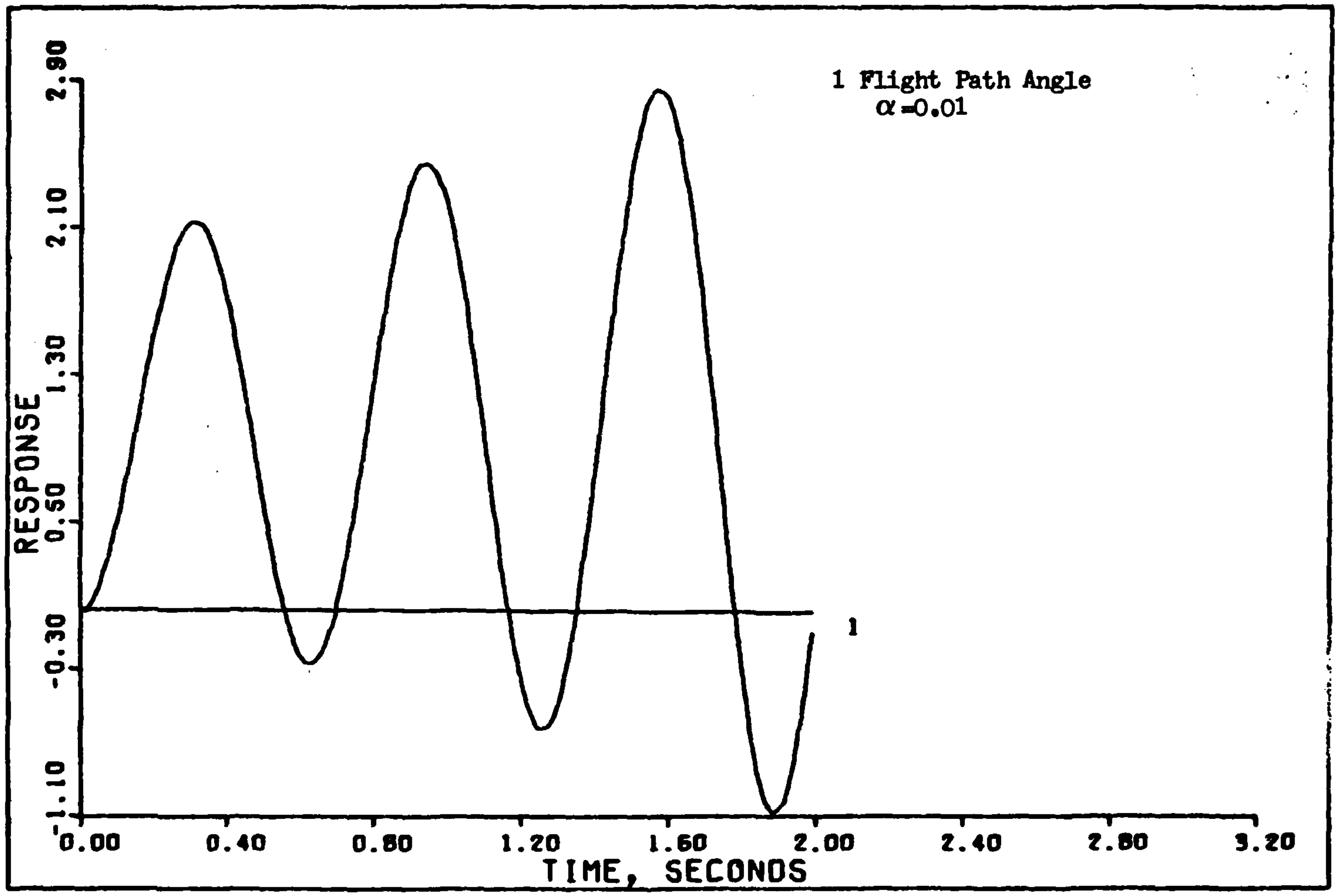


FIG. 18 MACH 0.18 FLIGHT CONDITION, FLIGHT PATH ANGLE COMMAND

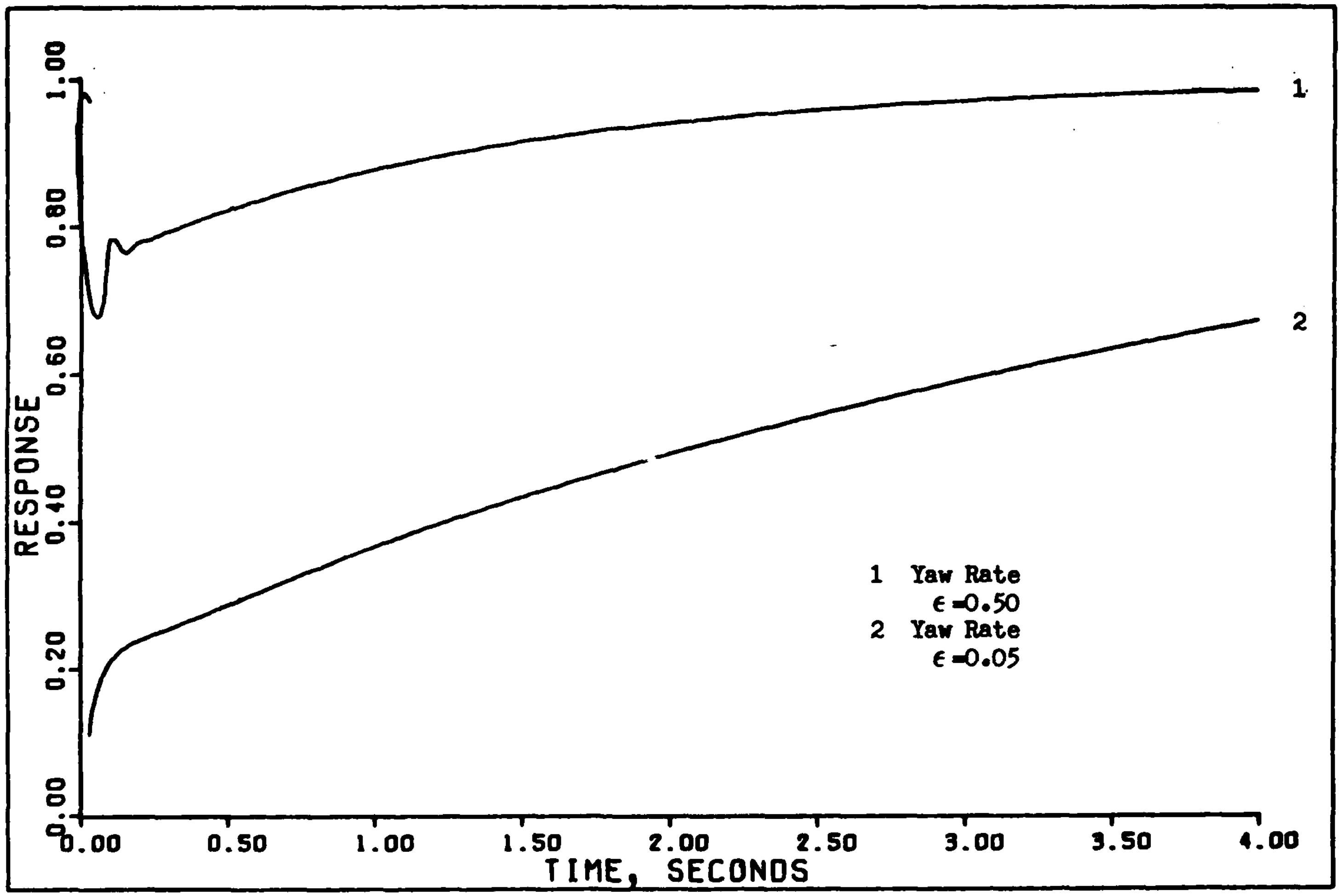


FIG. 19 MACH 0.18 FLIGHT CONDITION, YAW RATE COMMAND

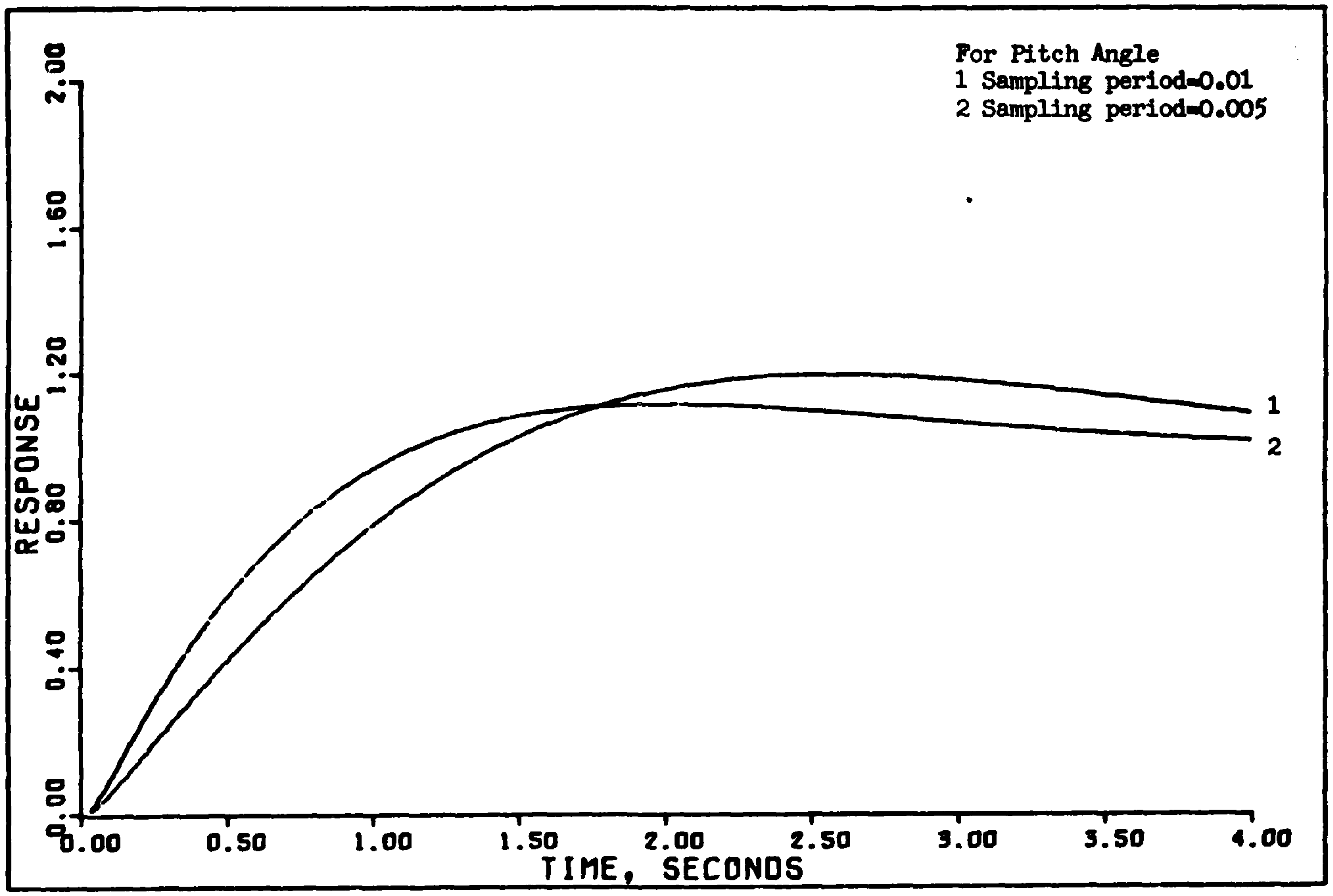


FIG. 20 MACH 0.18 FLIGHT CONDITION, PITCH ANGLE COMMAND

VII Conclusions and Recommendations

Thesis Summary

This thesis provides basic insight on the use of Professor Brian Porter's multivariable, digital control law design techniques to develop a robust, reconfigurable control law for the A-7D Digitac II aircraft.

Chapter I presents an introduction to the area of study. Chapter II describes the aircraft equations of motion and stability derivatives, and includes a discussion of the aircraft model for the A-7D aircraft. Chapter III presents a summary of the control law techniques for unknown, regular, and irregular plants. Chapters IV and V introduce the interactive computer program, MULTI. The actual control law development and analysis for the Mach 0.18 flight condition is given in Chapter VI.

The appendices of the thesis contain a MULTI User's Manual (Appendix A), a MULTI Programmer's Manual (Appendix B), and a program source listing (Appendix C).

The associate thesis by Paschall (Ref 3) should be reviewed by the reader who is interested in further details concerning the development of the aircraft equations of motion and stability derivatives. This reference also provides a listing of the state and output equations for the Mach 0.6 and 0.8 flight conditions, and a design and analysis of a control law for the Mach 0.6 case.

Conclusions

The research work of the thesis demonstrates that a tracker control law using the singular perturbation methods of Reference 6 can successfully be developed for the A-7D Digitac II aircraft. However, the final design has limitations and further work is required to improve the performance.

Although the overall tracking requirements of the control law can generally be met by adjusting the control law parameters, the high-gain aspect of the design requires flight control surface inputs which exceed the feasibility of the aircraft design. The control law presented in Chapter VI demonstrates robustness only for the forward velocity commands. Additional refinement of the controller design should be used to overcome these difficulties.

The first emphasis of the thesis work was the development of a fully operational computer-aided design program. This requirement severely limited the amount of mathematical research of the control law equations and the number of design trials which could be accomplished in the available time constraints.

It is also pointed out in Chapter VI that for some of the commanded inputs, the left and right horizontal stabilators showed non-symmetric deflections. This can be interpreted as a modeling problem, and the thesis research was terminated before these modeling problems could be further investigated.

Undesirable interaction between the states was also found for sideslip angle and roll angle commands. An aircraft can have a yaw angle and a yaw rate with zero sideslip; however, if there is an angle of sideslip there must be a yaw angle (Ref 18). For the control law design of Chapter VI, the A-7D Digitac II aircraft would seem to need a stronger yaw orientational control system, although this interaction might also be eliminated by adjusting the model so that the horizontal stabilator movements are symmetrical.

The MULTI computer program developed in this thesis is available from the Electrical Engineering Department, Air Force Institute of Technology and the Air Force Wright Aeronautical Laboratory (FIGL), Wright-Patterson AFB, Ohio. As the computer program is developed specifically for use in thesis research, it may require modification to be fully generic for use by the engineering community.

Recommendations

The conclusions presented in the previous section lead to the following recommendations:

1. Aircraft design must include sufficient wind tunnel testing which would provide all control derivatives needed to develop an aircraft model based on split control surfaces. The aircraft model should be tested by established methods before being used in any theoretical research.

2. A full mathematical analysis should be accomplished in an attempt to tabulate the effects of changes in the α , ϵ , and σ_i variables as related to the overall control law equations for unknown, regular, and irregular plants. This analysis would help reduce the number of "trial and error" simulations needed when designing a control law for an aircraft.

3. With the availability of the MULTI program, the design of controllers for the case of aircraft surface failures can be studied.

4. Future modifications to MULTI should include the capability for input command vectors which are "ramped-up", in addition to the current step input. The program should also be tested with the IMSL Library subroutine DVERK (Ref 14) replacing the ODE subroutine. The tests should check for increased accuracy and speed using the Runga-Kutta method for solution of differential equations.

5. The MULTI computer program should be modified to include a batch capability to take advantage of increased job time and core memory allocations.

Bibliography

1. Rubertus, Duane P. Handout on proposed AFIT thesis topics in EE 6.98. Wright-Patterson AFB, Ohio: Flight Dynamics Laboratory, January 1981.

2. Lipari, Louis J. DIGITAC II Phase I, Yaw Control Laws and Analytical Redundancy Evaluation. AFFDL-TR-79-5. Wright-Patterson AFB, Ohio: Air Force Flight Dynamics Laboratory, April 1979.

3. Potts, David W. Direct Digital Design Method for Reconfigurable Multivariable Control Laws for the A-7D Digitac II Aircraft. MS Thesis. Wright-Patterson AFN, Ohio: Air Force Institute of Technology, December 1980. (AFIT/GE/EE/80D-36).

4. Porter, B. Design of Set-Point Tracking and Disturbance-Rejection Controllers for Unknown Multivariable Plants. Progress Report for Period 15 November 1980 to 31 December 1980. Wright-Patterson AFB, Ohio: Air Force Wright Aeronautical Laboratory (FIGL), January 1981. (USAME/DC/101/81).

5. Bradshaw, A. and B. Porter. "Singular Perturbation Methods in the Design of Tracking Systems Incorporating Fast-Sampling Error-Actuated Controllers," International Journal of Systems Science, 12(10): 1181-1191 (October 1981).

6. Bradshaw, A. and B. Porter. "Singular Perturbation Methods in the Design of Tracking Systems Incorporating Inner-Loop Compensators and Fast-Sampling Error-Actuated Controllers," International Journal of Systems Science, 12(10): 1207-1220 (October 1981).

7. Paschall, Randy N. "Design of a Multivariable Tracker Control Law for the A-7D Digitac II Aircraft." Unpublished MS Thesis. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1981.

8. Bender, M. and A. Wolf. Flight Test Evaluation of a Digital Flight Control System for the A7-D Aircraft Simulation Test Plan. Contract F3361-73-C-3098. Wright-Patterson AFB, Ohio: Aeronautical Systems Division, 15 February 1974.

9. McDonnell Douglas Corporation. The USAF Stability and Control Digital DATCOM. Volume I User's Manual. Wright-Patterson AFB, Ohio: Air Force Flight Dynamics Laboratory, 1976. (AFFDL-TR-76-45).

10. Bender, M., et al. Flight Test Evaluation of a Digital Multimode Flight Control System for the A-7D Aircraft. Volume I - Design, Analysis, and Ground Test Phase. Wright-Patterson AFB, Ohio: Air Force Flight Dynamics Laboratory, August 1975. (AFFDL-TR-7597).

11. LTV Vought Aeronautics Division. A-7 Aero-Dynamic Data Report. Report Number 2-53310/5R-1981, 21 May 1965.

12. Hemami, A. PACKAGE 200. Program for Computing the Transient Behavior of Tracking Systems Incorporating Unknown Plants, Actuator and Sensor Dynamics and Digital Controllers. University of Salford, England: Department of Aeronautical and Mechanical Engineering, 1981.

13. Yourdon, E. and L. L. Constantine. Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design. New York: Yourdon, Inc., 1975.

14. International Mathematical and Statistical Libraries, Inc. IMSL Library 3, Edition 6, (FORTRAN) CDC6000/7000, CYBER 70/170 Series. Texas: International Mathematical and Statistical Libraries, Inc., 1977.

15. Control Data Corporation. FORTRAN Version 5 Reference Manual. Revision C. California: Control Data Corporation, 1980.

16. Control Data Corporation. FORTRAN Extended Loader Reference Manual. California: Control Data Corporation, 1980.

17. Larimer, S. J. TOTAL - An Interactive Computer-Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. MS Thesis. Wright-Patterson AFB, Ohio: Air Force Institute of Technology, March, 1978.

18. Blakelock, J. H. Automatic Control of Aircraft and Missles. New York: John Wiley and Sons, Inc., 1965.

19. Kennedy, T. A. The Design of Digital Controllers for the C-141 Aircraft Using Entire Eigenstructure Assignment and the Development of an Inter-Active Computer Design Program.

20. Aerospace Systems Division. ASD Computer Center INTERCOM Guide. Revision A. Wright-Patterson AFB, Ohio: ASD Computer Center Open Shop, September 1976.

21. Nikolai, P. J. and D. S. Clemm. Solution of Ordinary Differential Equations on the CDC 6600/CYBER 74 Processors. Technical Memorandum. Wright-Patterson AFB, Ohio: Air Force Flight Dynamics Laboratory, January 1977.

22. Aerospace Systems Division. ASD Computer Center CALCOMP Plotter Guide. Revision E. Wright-Patterson AFB, Ohio: ASD Computer Center Open Shop, March 1980.

23. Stamm, Maj M. R. Computer generated handout on FORTRAN V version of subroutine PLOTIT. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1981.

Appendix A: User's Manual for Program MULTI

Overview of MULTI

MULTI is an interactive program which enables a computer user to design and simulate digital, multivariable control laws for discrete, linear, multiple-input, multiple-output systems. The three design methods incorporated in this program were developed by Professor Brian Porter of the University of Salford, England (Refs 4; 5; and 6). To help the user become quickly familiar with the program, the following overview is provided.

MULTI is able to design and simulate control laws for three types of plants:

(1) Unknown — The plant state and output equations are unknown, but the steady-state transfer function matrix, $\underline{G}(0)$, is obtainable from off-line tests. This method is also applicable to known plants, if $\underline{G}(0)$ has full rank.

(2) Regular — The linear, multivariable plant dynamics are described by the usual state and output equations, and the first Markov parameter, $[\ \underline{C}_2\ \underline{B}_2\ ]$, has full rank.

(3) Irregular — The plant dynamics are described by state and output equations, and the first Markov parameter does not have full rank. Thus, the system must be augmented by a measurement matrix so that a control law may be developed.

All control law designs may be evaluated by a discrete-time simulation of the system, actuator, and sensor equations. The user may elect to obtain a terminal plot or a CALCOMP plot of the system input and/or system output responses.

MULTI focuses on Porter's digital design as opposed to a continuous design method. At various points during the design and simulation process, the program can provide intermediate information about system matrices and input/output values.

Input, design and simulation data is automatically stored on separate memory files for future use. This prevents the time consuming task of re-entering data when the same system is under study.

The program performs full error detection and diagnostics when there are deficiencies in input data, design data, plotting data, or if the plant equations are not sufficient for the type of design being tried. Input error detection and recovery is limited, relying mainly on the automatic CYBER error detection capability. The user should be wary of entering values inconsistant with the limitations described in this guide, since exceeding these limits may cause unexpected termination of the program. If this occurs the memory files are not generated and all data is lost.

It must be stated that MULTI is not as dynamic in its ability to receive input data as other programs with which the user may be familiar (i.e. TOTAL (Ref 17), CESA (Ref 19)). At present, character inputs to display current matrix values or system design values are not available except by requesting the proper option number. Also at present, entering a "$" symbol, rather than actual data, to abort an option, is not recognized unless specifically noted in the program instructions below. During data input, if a comma is entered rather than an actual data value, the previously entered data value is retained.

The "*" designator in the equations of this guide denotes multiplication.

1. Introduction to MULTI

The MULTI computer package contains approximately 70 ordered options which give the user an interactive, iterative approach in the design and simulation of control laws for linear, multivariable plants. A desirable control law assures that, for constant commands, the output tracks the input and that disturbance rejection is accomplished. The method can be used only if (1) the introduction of integral action preserves stabilizability, (2) the number of outputs is equal to or less than the number of inputs, and (3) the forward transfer function has no zero-valued transmission zeros.

Once the model is entered and the parameters for the control law are selected, a simulation can be run which provides a full time-sequential listing of input/output values and a tabular listing of the same data. The tabular listing is then used to obtain plots of the input/output values, if the user desires. The following discussion should give the user all the information for optimal use of the computer program package called MULTI.

To attach and run MULTI, after LOGIN, the user enters:

```
COMMAND-  CONNECT,INPUT,OUTPUT
COMMAND-  ATTACH,MULTI,ID=AFIT,SN=AFIT
COMMAND-  SCREEN,FULL
COMMAND-  (ATTACH DATA FILES)
COMMAND-  MULTI
```

The ID and SN entries will depend on local directives. MULTI is terminated by entering:

```
OPTION, PLEASE> #99
```

1.1 MULTI's Input Modes

MULTI has three input modes in which it requests input information from the user. These modes are called the OPTION Mode, the DATA Mode, and the QUERY Mode. Each mode has its own restrictions on allowable input and its own method of requesting information.

OPTION Mode. The OPTION Mode is the executive command mode for MULTI. It has the following prompt message:

```
OPTION, PLEASE > #
```

Once the program is in this mode, the user is allowed to select any option referring to input, design, simulation, plotting, or program termination. This mode is the main input mode of MULTI since it allows the user complete access to the program and data files.

DATA Mode. MULTI enters the data mode when information is needed to perform an option. Each data input request has a specific statement associated with it, and each request is terminated by the symbol, " > ". Normally, data input must be numerical and separated by either a comma, a space, or the return key of the terminal. There are other miscellaneous data mode options which require character entries. All replies can be accomplished by a single character except when the program asks the user to enter the choice of "INPUT" or "OUTPUT". In this case the program requires the entire word, spelled correctly, to continue properly.

QUERY Mode. MULTI uses the QUERY Mode as an input checking tool and to suppress data printing. The name QUERY Mode suggests that the program is asking the user if data is correct or is to be printed.

The messages in this mode are either of the form:

```
ENTER "0" TO SKIP DATA PRINTOUT
ENTER "1" TO OBTAIN DATA PRINTOUT...>
```

or of the form:

IS THIS CORRECT...YES,NO,$...>

In the latter case a "NO" reply will return the user to the data input point while a "$" reply will terminate the option. Although the option is terminated, the values just entered <u>are</u> placed in memory. The user can also enter a simple "Y" or "N".

1.2 <u>MULTI's</u> <u>Options</u>

MULTI contains 40 main options that allow the user versatility in the input, design, simulation and analysis of control laws for unknown, regular, or irregular plants. The options are grouped together into four major option blocks as follows:

BLOCK 1	OPTIONS # 0 thru # 9	Plant Input Options
BLOCK 2	OPTIONS # 10 thru # 19	Design Parameter Options
BLOCK 3	OPTIONS # 20 thru # 29	Simulation Options
BLOCK 4	OPTIONS # 30 thru # 39	Plotting Options

In addition, OPTIONS #100 thru #130 form Option BLOCK 5 which is composed of printing commands used to print the data entered into the program from the related main option. That is, to print the data which was entered into the program by using OPTION #5, the user selects OPTION #105.

Although the selection of a sampling time is actually a design parameter, BLOCK 2 is used only to form the control law matrices which are independent of sampling time. Thus, the sampling time selection is accomplished in the simulation option block where the actual control law is used.

2. Complete Description of MULTI's Options

In order to utilize MULTI's assets fully, it is necessary to have a complete understanding of the 40 main options available in MULTI. This manual is intended to provide all of the information needed to accomplish this requirement. The options described in the following sections may be selected by simply entering the option number while the program is in the option mode.

2.1 BLOCK 1 — Plant Input Options

Since a control law design cannot be started without the knowledge of the plant model or steady-state transfer function, it is necessary to enter some representation of the system for which the control law is to be designed. Thus, BLOCK 1 is an integral part of the MULTI program. Plant data may be entered via the $\underline{G}(0)$ matrix (OPTION #1) or by entering the individual matrices of the state-space representation of the plant (OPTIONS #2 and #3).

Although the steady-state transfer function, $\underline{G}(0)$, can be used to obtain control law matrices, it does not provide sufficient information to run a simulation and obtain plots of the responses by using option BLOCKS 3 and 4.

The state and output equations of the plant follow the form:

$$\underline{\dot{x}}(t) = \underline{A}\,\underline{x}(t) + \underline{B}\,\underline{u}(t) \qquad \text{(A-1)}$$

$$\underline{y}(t) = \underline{C}\,\underline{x}(t) + \underline{D}\,\underline{u}(t) \qquad \text{(A-2)}$$

where

$\underline{A}$	=	continuous-time plant matrix
$\underline{B}$	=	continuous-time control matrix
$\underline{C}$	=	continuous-time system output matrix
$\underline{D}$	=	continuous-time feed-forward matrix

Note that there is no provision for a disturbance matrix in Eq. (A-1).

The plant input option block also allows the user to input actuator and sensor state equations into the program. These equations have the same format as shown for the system state equations, however there is no feed-forward matrix, $\underline{D}$. If actuator and sensor values are not entered by the user, they are set equal to single-order servos with extremely fast time constants.

The program accepts a maximum of ten states, ten inputs, and ten outputs. The program only allows for 2nd-order actuators and sensors, thus, they may have to be approximated by 1st-order or 2nd-order equivalent servos.

<u>OPTION #0 — List Options 0 thru 9</u>

This option lists the plant input options from 0 to 9.

<u>OPTION #1 — Enter G(0) Matrix</u>

This option enters the steady-state transfer function matrix, $\underline{G}(0)$. The user is asked to supply the number of inputs, M, and the number of outputs, P, thus setting the dimension of the matrix. The data is then entered by row following each prompt message.

<u>OPTION #2 — Enter Number of States, Inputs, and Outputs</u>

This option asks the user to enter the number of states, N, the number of inputs, M, and the number of outputs, P. These values must be entered sequentially as N, M, and P.

<u>OPTION #3 — Enter A, B, C & D Plant Matrices</u>

This option enters the plant $\underline{A}$, $\underline{B}$, $\underline{C}$, and $\underline{D}$ matrices. Each input has the same format and the user enters the data by row after the prompt.

After each matrix is entered, its values are automatically echoed back to the user for checking. After the A, B, and C matrices are entered, the program asks if there is a D matrix. If the reply is negative, the option is terminated, and the program sets the D matrix values to zero.

OPTIONS #4 and #5 — Enter Actuator/Sensor State Equation Matrix Data

These options enter or eliminate the actuator and sensor state equation A, B, and C matrices. When these options are selected the following prompt will appear:

```
ENTER "0" TO ELIMINATE ACTUATORS/SENSORS
ENTER "1" TO SET ACTUATOR/SENSOR VALUES...>
```

If "1" is entered, the user is first asked to enter the number of states of each actuator/sensor. The input sequence must correspond to the sequence of the actuators/sensors. The user then inputs the A matrix values by row, the B column values and the C row values after each prompt.

OPTIONS #6 and #7 — Reserved Options

OPTION #8 — Copy G(O) Data from Local File

An alternate data input mode for OPTION #1 is to use OPTION #8. The data may be contained in any local file and need not contain *EOR and *EOF statements. The file should not be a permanent file called MEMO and if it is a local file by that name, its contents will be overwritten upon normal program termination. The file must be in the proper format as shown below:

```
100= 2                       Indicates G(O) information
110= 3 3                     M,P values
120= -1. 5. 0.      )
130= 1. 2. 7.       }        G(O) matrix values by row
140= 1.2 4. .123E01 )
```

The matrix data values can be entered in any format (i.e. real, integer, etc.). The line numbers above indicate that the data file may be created in the CYBER "EDITOR" (Ref 20), however the line numbers must be suppressed when the file is saved into a local file. For example, to save data values into a file named DATA1, the EDITOR command is:

```
..SAVE,DATA1,N
```

The numbers in each line of the data file may also begin in column one and may be separated by commas.

When using this option, the program asks the user to specify and verify the name of a local file which holds that data. The data is then read into memory and the user can verify the data entries by using OPTIONS #101 and #102.

OPTION #9 — Copy Plant, Actuator and Sensor Info from Local File

The option copies plant state equations, actuator and sensor information from local data file into computer storage locations. The file name restrictions and procedures mentioned in the OPTION #8 discussion also apply to this option. This option is the alternate to OPTIONS #2 through #5. The following lines show the exact format for the data file:

```
100= 1                        Indicates state equation information
110= 3 2 2                    N,M,P values
120= 1. 5.5 7E-02 )
130= .2 -1.2 0    }           A matrix values by row
140= .5 .005 99.1 )
150= 4. 0. )
160= 1. 1. }                  B matrix values by row
170= 0. 1. )
180= 0. 1 1. }
190= 1. 1. 1.}                C matrix values by row
200=N                         Indicates no D matrix
210=N                         Indicates no actuators
```

220=Y	Indicates sensor data follows
230= 1 2	# of states of sensor #1 and #2
240= -9999.	Sensor #1 $\underline{A}$ matrix value
250= 9999	Sensor #1 $\underline{B}$ matrix value
260= 1.	Sensor #1 $\underline{C}$ matrix value
270= 0. 1.	Sensor #2 $\underline{A}$ matrix values by row
280= -888.063 -147.2	
290= 0. 293.141	Sensor #2 $\underline{B}$ column values
300= 1. 0.	Sensor #2 $\underline{C}$ row values

The matrix data values can be entered in any format. The line numbers must be suppressed when the file is created in EDITOR and saved into a local file. The *EOR and *EOF entries, normally associated with a file, are not required.

When using OPTION #9, the program asks the user to specify and verify the name of a local file which holds the data. The data.is then read into memory and the user can verify the data entries, if desired, by using OPTIONS #102 thru #105.

2.2 BLOCK 2 — Design Input Options

MULTI's control law development is based upon forming the controller matrix, $\underline{K}$, from proper partitions of the $\underline{G}(0)$ or $\underline{A}$, $\underline{B}$, $\underline{C}$, and $\underline{D}$ matrices of the plant. In turn, $\underline{KO}$ and $\underline{K1}$ matrices are created using the scalar variables ALPHA and EPSILON. The equations are:

for unknown plants...

$$\underline{u}\,(kT) = T\,[\,\underline{KO} * \underline{e}\,(kT) + \underline{K1} * \underline{z}\,(kT)\,] \tag{A-3}$$

for regular or irregular plants...

$$\underline{u}\,(kT) = f\,[\,\underline{KO} * \underline{e}\,(kT) + \underline{K1} * \underline{z}\,(kT)\,] \tag{A-4}$$

where

$$\underline{KO} = \text{ALPHA} * \text{EPSILON} * \underline{K} \tag{A-5}$$

$$\underline{K1} = \text{EPSILON} * \underline{K} \tag{A-6}$$

$$f = 1 / T \tag{A-7}$$

In Eqs. (A-3) to (A-7),

$\underline{u}$ = input vector

$\underline{e}$ = proportional error vector

$\underline{z}$ = integral of error vector

T = sampling period

The $\underline{K}$ matrix in Eqs. (A-5) and (A-6) is of the form [$\underline{H}^{-1}$ * SIGMA] where SIGMA is a diagonal output weighting matrix, and

for unknown plants...

$$\underline{H} = \underline{G}(0) = \underline{D} - [\ \underline{C} * \underline{A}^{-1} * \underline{B}\ ] \qquad (A\text{-}8)$$

for regular plants...

$$\underline{H} = [\ \underline{C} * \underline{B}\ ] \qquad (A\text{-}9)$$

for irregular plants...

$$\underline{H} = [\ \underline{F}_2 * \underline{B}_2\ ] \qquad (A\text{-}10)$$

In the irregular case, the $\underline{F}_2$ matrix is the right partition of the augmentation matrix, $\underline{F}$, which is formed as follows:

$$\underline{F} = [\ \underline{F}_1\ ,\ \underline{F}_2\ ] = [\ \underline{C}_1 + \underline{M} * \underline{A}_{11}\ ,\ \underline{C}_2 + \underline{M} * \underline{A}_{12}\ ] \qquad (A\text{-}11)$$

The program refers to the matrix, $\underline{M}$, in Eq. (A-11) as the measurement matrix.

<u>OPTION #10 — List Options 10 thru 19</u>

This option lists the design parameter input options 10 to 19.

<u>OPTION #11 — Enter ALPHA</u>

This option is used to set the proportion of integral to direct feedback such that:

$$\underline{K0} = \text{ALPHA} * \underline{K1} \qquad (A\text{-}12)$$

OPTION #12 — Enter SIGMA Weighting Matrix

In this option the SIGMA weighting matrix diagonal elements are entered. This matrix is used in forming the controller matrix, $\underline{K}$, and has the effect of weighting the different inputs and outputs.

OPTION #13 — Enter EPSILON (SIGMA Matrix Multiplier)

The SIGMA matrix multiplier, ϵ , is entered using this option. EPSILON is used as shown in Eqs. (A-5) and (A-6).

OPTION #14 — Run Design...Unknown, Regular & Irregular Plants

In this option the $\underline{KO}$ and $\underline{K1}$ matrices are formed. The user is asked to enter the type of design to be accomplished and the reply can be either the full word or a single character (i.e. U, R, or I).

When using an unknown design with state equation input from OPTION #3, the user can opt to printout the $\underline{G}(0)$ matrix after it is formed.

If the regular design is used, the program asks if the user wishes to see the [$\underline{C}$ * $\underline{B}$] matrix after it is formed.

The first step in using the irregular design is to provide the program with a measurement matrix. If the measurement matrix is formed in OPTION # 18, previously entered in this option, or read in from a data file, the user need not reinitialize the values unless they are to be changed. The rows of the matrix are entered after the prompt is given. When all rows are received by the program, MULTI enters the QUERY Mode and requires the user to verify the values. When forming the $\underline{KO}$ and $\underline{K1}$ matrices, the user has the option of printing the [$\underline{F}_1$, $\underline{F}_2$] matrix.

When the matrices, $\underline{KO}$ and $\underline{K1}$, are formed, MULTI gives the user the message:

KO & K1 MATRICES FORMED

and the specific matrix values can be checked by using OPTION #114.

OPTIONS #15 thru #17 — Reserved Options

OPTION #18 — Measurement Matrix Formation... or Row & Column Operations on C * B or Other Matrix

This option has two functions as noted by the title.

In forming a measurement matrix to be used in the irregular plant controller design of OPTION #14, the user must enter the CESA feedback gain matrix, $\underline{K}_1$, for CESA Option #38 (Ref 19:124). MULTI advises when a measurement matrix has been formed. It can be printed by using OPTION #118.

This option can also be used to perform simple row and column operations on any matrix. There is provision for the user to perform these operations on the [$\underline{C}$ * $\underline{B}$] matrix without entering the actual matrix values. Otherwise the user must enter the matrix size and then the row values after each prompt. The matrix of interest can be no larger than 10x10. The addition/subtraction and multiplication/division operations are defined as follows:

enter X, Y, and Z such that...

$$X * Y + Z = Z' \qquad (A\text{-}13)$$
$$X * Y = X' \qquad (A\text{-}14)$$

where

X =	row or column number
X' =	modified row or column entered into matrix
Y =	multiplication/division factor
Z =	row or column number
Z' =	modified row or column entered into matrix

After each operation the resulting matrix is printed so that the next operation can be determined. The final matrix is not stored into any memory location for future use in other options and is destroyed at the termination of the option.

OPTION #19 — Copy Design Parameter Data from Local File

This option is used as an alternative to OPTIONS #11 thru #14 and copies design parameter data from a local data file. As it is mandatory that the data file be in the proper format, the following example file is provided:

100=U or R or I	Indicates Unknown,Regular,Irregular
110= 2	ALPHA value
120= .1 1 1	SIGMA matrix diagonal elements
130= .5	EPSILON value
140= 4.6 4.6 4.6	
150= -4.6 -4.6 -4.6	K0 matrix values by row
160= 1. 1. 6.	
170= 2.3 2.3 2.3	
180= -2.3 -2.3 -2.3	K1 matrix values by row
190= .5 .5 3.	
200= .25	
210= 0.	Measurement matrix values by row
220= 0.	

If the ALPHA value is 1, the file must not contain K1 matrix values. The measurement matrix values are read only if the file contains data for an irregular plant design. The line numbers above indicate only that the data file may be created in the CYBER "EDITOR". All file restrictions and procedures of OPTION #8 apply (substitute MEM10 for MEMO in OPTION #8 text).

Prior to using OPTION #19, either OPTION #2 or OPTION #9 must be used to set the values for the number of inputs and outputs. To use OPTION #19, the user must supply and verify the name of the local file which holds the design parameter data. The program reads the data into the proper memory locations and the user can verify the values by choosing OPTIONS #111 to #114 and OPTION #118.

2.3 BLOCK 3 — Simulation Options

The simulations options of this block are used to evaluate the controller matrices designed in OPTION #14. The simulation is performed after the user provides the initial values of the states and integrators, the command input vector, and the necessary time parameters. The actual simulation run is accomplished in OPTION #26. The simulation is obtained by solving a set of ordinary differential equations formed from the plant, actuator, and sensor state equation matrices and is run from time zero.

The ASD Library subroutine ODE (Ref 21) is the basis for the simulation. The user should be familiar with this subroutine, or as a minimum, have access to ODE's error descriptions.

During the simulation the control input is held constant over each sampling period. Since it may be advantageous to observe the system output between sampling periods, the user can specify a step time less than the sampling time.

As the simulation is run, a time-sequential printout of input and output values for every time interval of the simulation run can be selected. This printout includes actual input and controlled input values if the user has limited the inputs to certain values. If limitations are applied to the input values, the program generates a tabular listing of how often the limits are exceeded during the simulation. The simulation data is then stored in a matrix for later printout or for plotting.

OPTION #20 — List Options 20 thru 29

This option lists the simulation options from 20 to 29.

OPTION #21 — Set State & Integrator Initial Values

In this option the user specifies the state and integrator values at time $t = 0$. At present, all simulations must begin at zero time and proceed forward. The initial condition values for the states are entered after the first prompt message, and the integrator values are entered after the second prompt message. The values for each must be sequential.

OPTION #22 — Set Input Command Vector, V

This option is used to select the specific magnitude for each of the commanded inputs. At present, all inputs are step values. The entries for the input command vector are made sequentially.

OPTION #23 — Enter Sample Times

The sampling times are entered in this option. There may be up to two different sampling times entered. The program first asks how many sampling times will be entered, and then prompts the user to enter the same number of sampling time values.

OPTION #24 — Enter Simulation Time, TT

OPTION #24 is used to enter the time length for the simulation run. The user is cautioned to select this time with care since the combinations of simulation time/sample time or simulation time/step time determines the solution interval time. If the simulation time is too large, subroutine ODE may not be able to finish a simulation run before the CYBER CP time limit. In this case, MULTI is terminated without warning and no memory files are generated.

AD-A118 134   AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 1/3
DESIGN AND ANALYSIS OF A MULTIVARIABLE, DIGITAL CONTROLLER FOR --ETC(U)
DEC 81 D S PORTER
UNCLASSIFIED AFIT/GE/EE/81D-48 NL

2 of 3
AD
A118134

OPTION #25 — Enter Calculation Step Size, ST

As mentioned in the introduction to this section, the system input or output values between sampling times may be of interest. This option allows the user to select the step time between each sampling period. If the step time is chosen larger than the sampling time, the program proceeds using the sampling time as the discrete time interval.

OPTION #26 — Run Simulation

When OPTION #26 is chosen, MULTI forms and solves the set of ordinary differential equations formed from the plant, actuator and sensor state equations. Very fast actuators and sensors are approximated if no actual values have been entered, and the user receives messages to indicate that no values have been set. The user also receives a message that no control limits have been applied to the inputs, if such is the case.

MULTI runs a simulation for each sampling time selected in OPTION #23. As the program proceeds through each sampling time, the user is first given the length of the run, the step time and the sampling time. The user is then told how many time increments are generated during the simulation run. The user can opt to suppress the time-sequential listing of input/output values. The message:

```
-----CALCULATIONS IN PROGRESS-----
```

is provided if the time-sequential listing is suppressed, since the solution, in some cases, may take several seconds.

If the number of time increments over the simulation time interval is greater than 100, the input/output data must be "packed" into a matrix for later use in plotting or providing a tabular listing of the input/output values. When this occurs, missing time increments are noted when the tabular data of input/output values are listed. However, the user can always obtain a full list of these values by choosing a time-sequential listing.

At the conclusion of each simulation run, the program provides a table telling how many times the input values exceeded the limitations chosen in OPTION #27. This listing is not retained in memory and can be accessed only once.

The tabular listing of the input/output data points for each sampling time can be obtained from OPTION #126.

OPTION #27 — Set Control Input Limits

In this option the user can set input control limitations. The program supplies the correct directive on how to enter the limits and provides proper messages when the limitations are to be entered. The user enters the number corresponding to the input which is to be limited, and then gives the minimum and maximum values of the limitation.

This option can also be used to eliminate control input limitations if they are previously entered.

To obtain a listing of the current input limits, OPTION #127 is used.

OPTION #28 — Reserved Option

OPTION #29 — Copy Simulation Parameters for a Local File

This single option is used to enter state and integrator initial values, command vector values, sampling times, simulation run length and interval step time. It can be used as an alternate to OPTIONS #21 through #25. The reader should refer to the discussion of OPTION #8 for additional information relating to file structure and related procedures (substitute MEM20 for MEMO in OPTION #8 text).

The data file must follow the format shown below:

100= 0. 0. 0.	Initial state values
110= 0. 0.	Initial integrator values
120= 1. -1.	Command vector values (transposed)
130= 2 .01 .02	# of sampling times, sampling periods
140= 3.	Simulation run time
150= .01	Step time

As with the other data file examples in OPTION #8, OPTION #9, and OPTION #19, the line numbers above only indicate that the data matrix may be created in EDITOR, and must be suppressed when the file is saved into a local file. The *EOR and *EOF terminators are optional.

Prior to using this option, the number of states, inputs and outputs must be set by using OPTION #2 or OPTION #9. To use OPTION #29 the user must enter and verify the name of the local file which holds the simulation data. After the program reads the data into memory, the user can verify the entries by using OPTIONS #21 to #25.

2.4 <u>BLOCK 4</u> — <u>Plotting Options</u>

An integral aspect of any design is the ability to quickly interpret and analyze the results. The quickest method of analyzing results is to see them in either tabular or graphical form. MULTI provides the user with two types of graphical interpretation tools.

The first and fastest tool is a quick sketch at the user's terminal of any combination of input/output responses. The second tool is a CALCOMP plot. Each has its own advantages and limitations. The use of both is explained in this section.

To provide a plot, MULTI requires several questions to be answered concerning what data is to be plotted. OPTIONS #31, #32, and #33 produce terminal plots, and OPTIONS #34, #35, and #36 produce CALCOMP plots. After using OPTION #31/OPTION #34, the user becomes aware that an easier method of requesting plots is required. In this light, a shortened version of requesting plots is available by using OPTION #32/#35. Finally, if the user desires the same information to be plotted after another simulation (i.e. output #2 vs. input #3) and has already entered these choices by using OPTION #31 or #32/OPTION #34 or #35, an immediate plot can be generated by using OPTION #33/OPTION #36.

OPTION #30 — List Options 30 thru 39

This option lists the simulation options from 30 to 39.

OPTIONS #31 thru #36 — Terminal Plots and CALCOMP Plots

When OPTION #26 has been completed, MULTI has formed and stored a three-dimensional matrix of data values. For this discussion it is only necessary to explain that the row dimension is equivalent to the input or output values at each time increment, the column dimension is equivalent to the input or output number, and the third dimension is equivalent to the sampling time. A plot is generated after a user chooses the type of plot to be generated, the sampling time of interest, input or output, the number of inputs or outputs and the particular inputs or outputs of interest. After this information is received, MULTI can proceed with plotting the correct data matrix rows and columns.

There are four basic types of plots that can be obtained:

for a single sampling time...

1...a plot of up to 2 input and output pairs
2...a plot of up to 4 inputs or outputs
3...a plot of up to 4 different simulations
(for any single input or output

or for up to 4 different sampling times...

4...a plot of any single input or output

For the third type of plot, the user returns to the main program to run another simulation and then re-enters the plotting routine. The plot is generated after all simulations are run. When planning to plot data for two different sampling times, it is mandatory that the step time, for the simulation runs, be equal to or less than the smallest sampling time. The simulations must also be run for the same length of time. The program provides a warning indicating these restrictions. If these restrictions are not followed, MULTI is terminated early and no data files are generated by the program.

OPTIONS #31 through #33 pertain to terminal plots, or quick sketches at the user's terminal. OPTIONS #34 through #36 are for CALCOMP plots. As mentioned above, there are shortened versions of both the terminal plot and CALCOMP plot requests. The same plots are obtained with these shortened versions as are produced by using the full versions. The difference lies in entering the data for the plot.

The shortened versions included in OPTIONS #32 and #35 merely suppress the informational prompts that are available with OPTIONS #31 and #34. The user simply enters the plotting type, input or output, and the needed plotting data for the plotting type chosen. These two options are not recommended for the inexperienced MULTI user.

If using these two options, the needed plotting data is as follows:

Type #1 — Input/Output Pairs

ENTER...# of pairs (OPTION #35 only), sampling time #, input #s, output #s

Type #2 — Inputs or Outputs

ENTER...sampling time #, # of inputs/outputs, input/output #s

Type #3 — Multiple Simulations

ENTER...# of simulations, sampling time #, input/output #

Type #4 — Multiple Sampling Times

ENTER...# of sampling times, sampling times, input/output #

Once the information for a plot has been entered by using OPTIONS #31, #32, #33 or #34, the user can return to the program to change parameters and run a new simulation, and then choose OPTIONS #33 or #36 to obtain a plot. This plot of the new simulation data is a plot of the same type as previously requested.

The CALCOMP plots are stored in a local file called PLOT which the user can route to the plotter after MULTI is terminated (see ..ction 2.6). Local restrictions permit no more than five plots on one plotting file, thus, MULTI advises the user when five plots have been generated. All CALCOMP plots are bordered by a rectangular box which is 6x9 inches in size. The plotting size factor is entered as "1" for this size of CALCOMP plot, but the size can be varied by changing the size factor. The user is also required to provide two titles for the plot. The X-axis is always labeled "TIME"; the user selects the title to be placed along the Y-axis of the plot. The Y-axis title can be no more than 30 characters in length. The user must also provide an overall title for the plot which is drawn below the lower nine inch side of the rectangle. This main title can be up to 60 characters long. For both title entries, the user should center the plot title between the prompts so that the titles will be centered on the CALCOMP plot:

> center title between prompts <

<u>OPTIONS #37 thru #39 — Reserved Options</u>

2.5 <u>BLOCK 5</u> — <u>Printing Options</u>

All available options in this block print data which the user has entered using the options in BLOCKS 1 to 4.

<u>OPTION #100 — List Options 100 thru 130</u>

This option lists the printing options from 100 to 130.

<u>OPTIONS #101 to #129 — Print Data Entered in OPTIONS #1 to #29</u>

As explained in Section 1.2, the options in this range are related to the sub-100 numbered options. To print out the data entered by OPTION #1, the user selects OPTION #101; to print out the data entered by OPTION #2, the user selects OPTION #102; etc.

OPTION #103 prints out the data from OPTION #2 and then gives the user the option of selecting the $\underline{A}$, $\underline{B}$, $\underline{C}$, or $\underline{D}$ matrix from OPTION #3.

ALPHA and EPSILON are combined in one printout and obtained by selecting OPTIONS #111 or #113.

OPTION #114 prints out the control matrices, $\underline{K0}$ and $\underline{K1}$, from OPTION #14 and also prints out a heading explaining which type of plant (unknown, regular, irregular) the user chose in generating the matrices. If this heading is suppressed, the program is unsure of how the matrices were generated, but the values displayed are still the current K0 and K1 matrices.

The data from OPTIONS #21 and #22 are also combined into one printout and the user obtains both sets of data when selecting OPTION #121 or OPTION #122.

The same is true for the sampling times, simulation time, and calculation step size. They form one printout and can be printed by selecting OPTION #123, #124, or #125.

When OPTION #126 is entered, the user is required to choose the tabular listing for the input or the output data. The full word, INPUT or OUTPUT, must be entered and spelled correctly for the program to continue properly.

OPTION #130 — List Plotting Selections (currently not available)

This single option is used to determine the current plotting selections from the last entry to OPTIONS #31 to #36. When chosen, the program tells the user if the data is for a terminal plot or a CALCOMP plot and then gives the current plotting choices.

2.6 MULTI Generated Memory Files

When OPTION #99 is used to end MULTI, there are three data files generated. A file called MEMO is formed from the data entered from option BLOCK 1; a file called MEM10 is created from the data entered from BLOCK 2; a file called MEM20 is opened to hold the data entered in option BLOCK 3. These files are local files and should be rewound using the command:

```
COMMAND- REWIND,(FILENAME)
```

before they are stored into permenent file space, edited, or cataloged.

Another file called PLOT is also generated if the user selects any of the CALCOMP plotting options. This file should also be rewound before it is routed to the plotter.

The routing command for MEMO, MEM10, and MEM20 is:

```
COMMAND- ROUTE,MEM#,DC=PR,TID=91,FID=XXXXX,ST=CSB
```

and the routing command for PLOT is:

```
COMMAND- ROUTE,PLOT,DC=PT,TID=91,FID=XXXXX,ST=CSB
```

The routing commands send the files to the facilities at AFIT. The user can choose any five letter "flag ID" designated above as "XXXXX".

3. Summary of MULTI's OPTIONS

MULTI contains four main option BLOCKS which have ten options each, and 30 printing options. The MULTI options are summarized on the computer listing that follows this section.

To obtain a controller design and simulation these option numbers, in the following order, are generally used:

for unknown plants...

1 (or 9); 11, 12, 13, 14U
or 2, 3 (or 9); 11 to 14U (or 19); 21 to 25 (or 29), 26; 31 to 36

for regular plants...

2, 3 (or 9); 11 to 14R (or 19); 21 to 25 (or 29), 26; 31 to 36

for irregular plants...

2, 3 (or 9); 11 to 14I (or 19); 21 to 25 (or 29), 26; 31 to 36

If a measurement matrix is to be generated for the irregular case, OPTION #18 should be used prior to OPTION #14. In all cases, OPTIONS #4, #5 and #27 (which set specific actuator, sensor and input control limits) should be used, if desired, prior to OPTION #26.

```
OPTION, PLEASE > #0

PLANT INPUT OPTIONS:
     0.  LIST OPTIONS 0 THRU 9
     1.  ENTER G(0) MATRIX
     2.  ENTER # OF STATES, INPUTS & OUTPUTS (N,M,P)
     3.  ENTER PLANT A, B, C, & D MATRICES
     4.  ENTER ACTUATOR STATE EQUATION MATRIX DATA
     5.  ENTER SENSOR STATE EQUATION MATRIX DATA
     6.  OPTION RESERVED
     7.  OPTION RESERVED
     8.  COPY G(0) INFO FROM LOCAL FILE
     9.  COPY PLANT, ACTUATOR & SENSOR INFO FROM LOCAL FILE


OPTION, PLEASE > #10

DESIGN PARAMETER INPUT OPTIONS:
    10.  LIST OPTIONS 10 THRU 19
    11.  ENTER ALPHA
    12.  ENTER SIGMA WEIGHTING MATRIX
    13.  ENTER EPSILON (SIGMA MATRIX MULTIPLIER)
    14.  RUN DESIGN...UNKNOWN, REGULAR & IRREGULAR PLANTS
    15.  OPTION RESERVED
    16.  OPTION RESERVED
    17.  OPTION RESERVED
    18.  MEASUREMENT MATRIX FORMATION...OR
         ROW & COLUMN OPERATIONS ON C*B OR OTHER MATRIX
    19.  COPY DESIGN PARAMETERS FROM LOCAL FILE


OPTION, PLEASE > #20

SIMULATION OPTIONS:
    20.  LIST OPTIONS 20 THRU 29
    21.  SET STATE & INTEGRATOR INITIAL VALUES, X(0) & Z(0)
    22.  SET INPUT COMMAND VECTOR, V
    23.  ENTER SAMPLE TIMES
    24.  ENTER SIMULATION TIME
    25.  ENTER CALCULATION STEP SIZE
    26.  RUN SIMULATION
    27.  SET CONTROL INPUT LIMITS
    28.  OPTION RESERVED
    29.  COPY SIMULATION PARAMETERS FROM LOCAL FILE
```

```
OPTION, PLEASE > #30

PLOTTING OPTIONS:
    30.  LIST OPTIONS 30 THRU 39
    31.  QUICK SKETCH AT USER'S TERMINAL
    32.  QUICK SKETCH--SHORT VERSION
    33.  QUICK SKETCH--RETAINING SAME PLOTTING CHOICES
    34.  CALCOMP PLOT
    35.  CALCOMP PLOT--SHORT VERSION
    36.  CALCOMP PLOT--RETAINING SAME PLOTTING CHOICES
    37.  OPTION RESERVED
    38.  OPTION RESERVED
    39.  OPTION RESERVED


OPTION, PLEASE > #100

ALL 100-SERIES OPTIONS PRINT DATA VALUES...
    FOR VALUES SET IN OPTION #1...USE OPTION #101
    FOR VALUES SET IN OPTION #2...USE OPTION #102
ETC.
    FOR PLOTTING SELECTIONS.......USE OPTION #130
```

4. Example for Irregular Plants

This manual is concluded with a computer generated design and simulation for an irregular plant. The plant is defined by the following matrices for the state and output equations:

$$\underline{A} = \begin{bmatrix} 0.0 & 1.1320 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & -0.1712 & -0.0538 & 0.0 & 0.0705 \\ 0.0 & 0.0 & 0.0485 & -0.8536 & -1.013 \\ 0.0 & 0.0 & -0.2909 & 1.0532 & -0.6859 \end{bmatrix} \qquad \text{(A-15)}$$

$$\underline{B}^T = \begin{bmatrix} 0.0 & 0.0 & -0.0012 & 0.4419 & 0.1575 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & -1.6646 & -0.0732 \end{bmatrix} \qquad \text{(A-16)}$$

$$\underline{C} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} \qquad \text{(A-17)}$$

and the following parameter values are used in the design:

ALPHA	= 1.0
SIGMA	= diag [1 , 1 , 1]
EPSILON	= 1.0
All Initial Values	= 0
Command Vector, V	= [1 , 0 , 0]
Sampling Time	= .02
Simulation Time	= 4.0
Calculation Step Size	= .02

The computer printout includes a terminal plot of output #1, #2, and #3. Note that when values are duplicated, the terminal plot symbols will overwrite each other (i.e. output #3 values overwrite output #2 values). The CALCOMP plot of the same information is included as Fig. A-1.

```
COMMAND- EDITOR
..C
   100=1
   110=5,3,3
   120=0,1.132,0,0,1
   130=0,0,0,1,0
   140=0,-.1712,-.0538,0,.0705
   150=0,0,.0485,-.8536,-1.013
   160=0,0,-.2909,1.0532,-.6859
   170=0,0,0
   180=0,0,0
   190=-.0012,1,0
   200=.4419,0,-1.6646
   210=.1575,0,-.0732
   220=1,0,0,0,0
   230=0,0,1,0,0
   240=0,1,0,0,0
   250=NO
   260=NO
   270=NO
   280==
..SAVE,EXAMPLE,N
..B,B
COMMAND- MULTI


WELCOME TO MULTIVARIABLE DESIGN
C 1981    PORTER,SMYTH,PASCHALL

THIS PROGRAM USES THE DESIGN TECHNIQUES DEVELOPED BY
PROFESSOR BRIAN PORTER, UNIV. OF SALFORD, ENGLAND


OPTION, PLEASE > #9

THIS OPTION COPIES STATE EQN INFO FROM A LOCAL DATA FILE

ENTER LOCAL FILE NAME THAT HOLDS STATE EQN INFORMATION
                   >EXAMPLE
LOCAL FILE NAME IS >EXAMPLE

IS THIS CORRECT...YES,NO,$...>YES

DATA COPY COMPLETE FOR OPTIONS
#2, #3
```

```
OPTION, PLEASE > #11

THIS OPTION SETS THE PROPORTION OF INTEGRAL AND DIRECT FEEDBACK

ENTER ALPHA >1


OPTION, PLEASE > #12

THIS OPTION SETS THE WEIGHTING BETWEEN OUTPUT CHANNELS

ENTER SIGMA WEIGHTING MATRIX...3 DIAGONAL ELEMENTS ONLY
DIAGONAL ELEMENTS >1,1,1


OPTION, PLEASE > #13

THIS OPTION SETS THE WEIGHTING MATRIX MULTIPLIER

ENTER SIGMA MATRIX SCALAR MULTIPLIER, EPSILON >1


OPTION, PLEASE > #14

ENTER DESIGN METHOD...UNKNOWN,REGULAR,IRREGULAR...>IRREGULAR

THIS OPTION COMPUTES K0 & K1 FOR IRREGULAR PLANTS

ENTER "1" TO REINITIALIZE MEASUREMENT MATRIX, ELSE ENTER "0" >1

ENTER M MATRIX...3 ROWS WITH 2 COLUMNS
ROW 1 >.25,0
ROW 2 >0,0
ROW 3 >0,.25

MEASUREMENT MATRIX...

   .2500E+00   0.
  0.           0.
  0.            .2500E+00


IS THIS CORRECT...YES,NO,$...>YES


ENTER "0" TO SKIP F=[F1,F2] MATRIX
ENTER "1" TO OBTAIN THIS DATA PRINTOUT...>1
```

```
F=[F1,F2] MATRIX...

   .1000E+01    .2830E+00   0.           0.            .2500E+00
  0.           0.             .1000E+01  0.           0.
  0.             .1000E+01  0.            .2500E+00   0.


KO & K1 MATRICES FORMED


OPTION, PLEASE > #114

CONTROL MATRICES ARE FOR PLANTS WHICH ARE IRREGULAR

KO MATRIX...

   .2897E+02   0.           -.1274E+01
   .3477E-01    .1000E+01   -.1529E-02
   .7691E+01   0.           -.2741E+01


K1 MATRIX IS IDENTICAL TO KO MATRIX



OPTION, PLEASE > #21

THIS OPTION SETS THE INITIAL CONDITION VECTORS FOR THE STATES & INTEGRATORS

ENTER THE X( 0 ) VECTOR OF 5 ELEMENTS
>0,0,0,0,0

ENTER THE Z( 0 ) VECTOR OF 3 ELEMENTS
>0,0,0


OPTION, PLEASE > #22

THIS OPTION SETS THE INPUT COMMAND VECTOR, V

ENTER THE V VECTOR OF 3 ELEMENTS
"V" COLUMN >1,0,0


OPTION, PLEASE > #23

THIS OPTION SETS THE SAMPLING TIME FOR EACH RUN

ENTER NUMBER ( MAX OF 2 ) OF SAMPLING TIMES >1
ENTER 1 SAMPLING TIME( S ) >.02
```

```
OPTION, PLEASE > #24

THIS OPTION SETS THE TOTAL SIMULATION TIME

ENTER TOTAL TIME >4


OPTION, PLEASE > #25

THIS OPTION SETS THE CALCULATION STEP SIZE

ENTER STEP SIZE >.02


OPTION, PLEASE > #26

THIS OPTION RUNS THE SIMULATION IF ALL DATA IS AVAILABLE

SIMULATION INCLUDES NO ACTUATORS
SIMULATION INCLUDES NO SENSORS
SIMULATION INCLUDES NO CONTROL LIMITS


RUN TIME=4.  STEP TIME=.02  SAMPLE TIME=.02

THERE ARE 200 TIME INCREMENTS
   ENTER "0" TO SKIP SEQUENTIAL LISTING
   ENTER "1" TO OBTAIN THIS DATA...>0

----CALCULATIONS IN PROGRESS----


SIMULATION FOR SAMPLING TIME #1 COMPLETE



OPTION, PLEASE > #31

THIS OPTION PRODUCES A PLOT AT YOUR TERMINAL

PLEASE CHOOSE ONE OF THE FOLLOWING:

FOR A SINGLE SAMPLING TIME
      1...A PLOT OF UP TO 2 INPUT AND OUTPUT PAIRS
      2...A PLOT OF UP TO 4 INPUTS OR OUTPUTS
      3...A PLOT OF UP TO 4 DIFFERENT SIMULATIONS
              (FOR ANY SINGLE INPUT OR OUTPUT)

OR FOR UP TO 4 DIFFERENT SAMPLING TIMES
      4...A PLOT OF ANY SINGLE INPUT OR OUTPUT

ENTER CHOICE DESIRED >2
```

```
CHOICE #2...YOU'VE CHOSEN TO PLOT INPUTS OR OUTPUTS

YOU MIGHT HAVE SELECTED TO RUN SEVERAL SAMPLING TIMES...
DESIGNATE THESE SAMPLING TIMES AS...1,2,3, ETC...
     ENTER THE 1 SAMPLING TIME(S) OF INTEREST >1

ENTER...INPUT OR OUTPUT...FOR YOUR PLOT >OUTPUT

HOW MANY OUTPUTS DO YOU WANT TO PLOT >3

WHICH OUTPUT(S) DO YOU WANT TO PLOT...
     ENTER THE 3 CORRESPONDING COLUMN NUMBER(S) >1,2,3

FOR NO GRID ON PLOT ENTER "0", FOR A GRID ENTER "1" >1



  1.05    -+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
           I          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  .939    -+     + XXXX+     +     +     +     +     +     +     +     +
           I     XXX                                                   I
  .827    -+     X     +     +     +     +     +     +     +     +     +
           I    X                                                      I
  .715    -+   X +     +     +     +     +     +     +     +     +     +
           I  X                                                        I
  .603    -+     +     +     +     +     +     +     +     +     +     +
           I  X                                                        I
  .491    -+     +     +     +     +     +     +     +     +     +     +
           I X                                                         I
  .378    -+     +     +     +     +     +     +     +     +     +     +
           I X                                                         I
  .266    -+     +     +     +     +     +     +     +     +     +     +
           IX                                                          I
  .154    -+     +     +     +     +     +     +     +     +     +     +
           I                                                           I
 4.228E-02-X     +     +     +     +     +     +     +   00+0    +0 0 0+
           *************************************************************
-6.979E-02-+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

            0.                                                 3.98


CURVE X ABOVE IS OUTPUT 1
CURVE O ABOVE IS OUTPUT 2
CURVE * ABOVE IS OUTPUT 3
```

```
OPTION, PLEASE > #35

THIS OPTION PRODUCES A CALCOMP PLOT

ENTER SHORT VERSION CHOICE...1,2,3,4...>2
ENTER...INPUT,OUTPUT...>OUTPUT
ENTER PROPER PLOT DATA...>1,3,1,2,3
ENTER TITLE FOR Y-AXIS OF PLOT (MAX OF 30 CHARACTERS)
              >          RESPONSE             <
TITLE IS:                RESPONSE

IS THIS CORRECT...YES,NO,$...>YES

ENTER MAIN PLOT TITLE (MAX 60 CHARACTERS)
              >           FIG. A-1    EXAMPLE -- IRREGULAR PLANT            <
MAIN TITLE:               FIG. A-1    EXAMPLE -- IRREGULAR PLANT

IS THIS CORRECT...YES,NO,$...>YES

ENTER PLOT SIZE FACTOR...>1.8

------------------------------------------------------------------------
LOCAL FILE "PLOT" CONTAINS THE CALCOMP DATA FOR THIS ENTRY TO OPTION #35
                YOU HAVE GENERATED A TOTAL OF 2 PLOT(S)
          BE SURE TO ROUTE "PLOT" TO THE PRINTER BEFORE LOGOUT
------------------------------------------------------------------------


OPTION, PLEASE > #99

ALL PLANT INPUT DATA HAS BEEN SAVED IN A LOCAL FILE
     CALLED "MEMO"

ALL DESIGN DATA HAS BEEN SAVED IN A LOCAL FILE
     CALLED "MEM10"

ALL SIMULATION DATA HAS BEEN SAVED IN A LOCAL FILE
     CALLED "MEM20"


HAVE A NICE DAY!

     STOP
      56600 MAXIMUM EXECUTION FL.
     18.655 CP SECONDS EXECUTION TIME.
COMMAND- REWIND,PLOT
COMMAND- ROUTE,PLOT,DC=PT,TID=90,FID=IRREG,ST=CSB
```

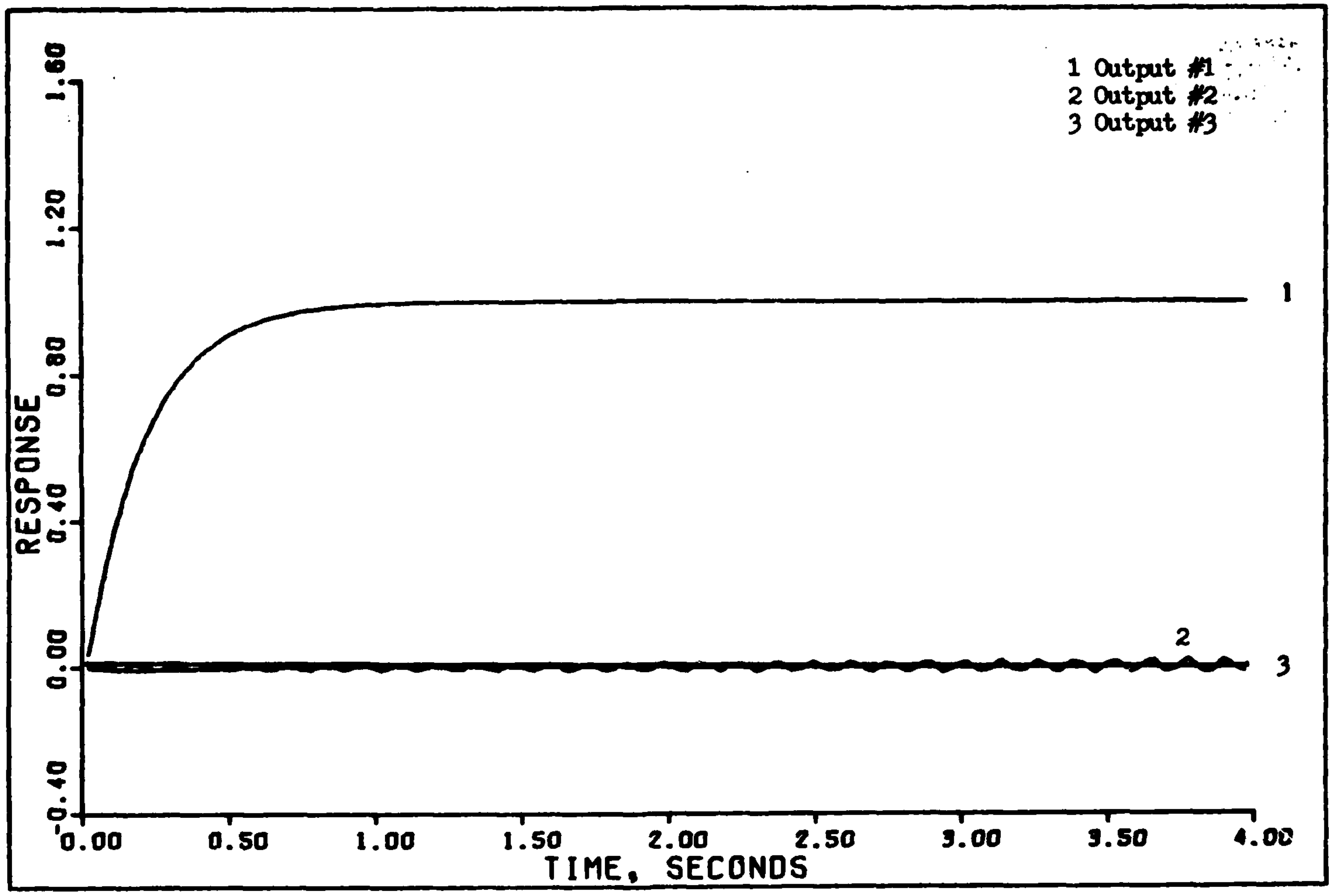


FIG. A-1 EXAMPLE -- IRREGULAR PLANT

Appendix B: Programmer's Manual for Program MULTI

1. Introduction

This guide provides the documentation needed for future modifications to MULTI. It is also intended to help a programmer analyze the flow structure of the program so that MULTI can be modified if unexpected errors occur. This manual describes the overlay structure, local and attached subroutines and the names of all elements of the program.

MULTI is written in FORTRAN V and is fully documented by COMMENT statements throughout the program. The program structure is such that programmers familiar with FORTRAN V can understand MULTI's operation.

The programmer should have a full, working knowledge of the theory behind the design of discrete-time, error-actuated controllers for linear, multivariable plants as developed by Professor Brian Porter, University of Salford, England (Refs 4; 5; and 6). The programmer should also have used MULTI interactively.

MULTI retains the code from the University of Salford which deals with measurement matrices for the $\underline{C}$ and $\underline{G}(0)$ matrices. These measurement matrices are not utilized in the current design code, and thus throughout MULTI, the following matrix identities exist:

$$\underline{CM}\,(I,J) = \underline{C}\,(I,J) \qquad (B\text{-}1)$$
$$\underline{GMO}\,(I,J) = \underline{GO}\,(I,J) \qquad (B\text{-}2)$$

The reader should obtain a MULTI source listing before continuing with this guide and have a CDC FORTRAN V Reference Manual (Ref 15) and a CDC Loader Reference Manual (Ref 16) available. A source listing is available from the Engineering Department at AFIT. Other programming guides which may be helpful to a potential MULTI programmer include:

IMSL Library Reference Manual (Ref 14)
ASD Computer Center CALCOMP Plotting Guide (Ref 22)
Solution of Ordinary Differential Equations on the CDC 6600/CYBER 74 Processors (Ref 21)
Computer Printout for Subroutine PLOTIT (Ref 23)

2. Description of Overall Structure

MULTI is written to provide a user with an interactive design tool for the design of control laws needed to attain tracking and disturbance rejection in a multi-variable plant. It is necessary for the program to have the capability of evaluating the input and output responses after the control law is simulated. In addition, it is necessary, for an iterative design approach, that the program retain all input data between designs. Finally, the program must be fully interactive.

MULTI must fulfill all these requirements and be able to operate in the limited $65{,}000_8$ words of memory core that is available of AFIT's INTERCOM. In its original form MULTI required in excess of $110{,}000_8$ words of memory. Thus it was necessary to redesign MULTI using an overlay structure and labeled common blocks.

2.1 Overlays

Overlays are used to reduce the storage requirements of large programs by dividing the program into modules. All modules are separate programs in their own right and are linked together by the use of a main executive module.

The main executive module and all common variables of the program constantly remain in operational core. The main executive directs the program flow by calling the primary overlays into operational core as they are needed.

The data from common variables is passed between the main and primary overlays by declaring the variables in COMMON statements in the main executive overlay.

The CDC FORTRAN V Reference Manual and the CDC Loader Reference Manual for NOS and NOS/BE can provide a programmer with more detailed information on overlays and COMMON statements.

2.2 MULTI's Overlay Structure

MULTI is composed of one main overlay and 13 primary overlays. The main overlay provides the executive directing function of the program. As an executive, the overlay initializes and stores data in specified common blocks. When an option number is entered by the user, the main overlay checks an IF/ELSEIF structure to attach the primary overlay needed to satisfy the option request. The main overlay also contains subroutines which are used by more than one primary overlay.

A main overlay description is given in Table B-I.

Initially there were three primary overlays; each was chosen to conform to three developmental aspects of controller design: data input, control law synthesis and simulation. However, to further reduce memory core requirements, the overlays and related subroutines are

TABLE B-I
Description of Main Overlay

MAIN OVERLAY			
Overlay #	Program Name	Options Included	
(0,0)	MULTI	All	
Overlay #	Subroutines		Purpose
(0,0)	MATPR		Prints matrices
	QPRINT		Asks of data is to be printed
	ANSWER		Asks if data is correct
	INVERT		Inverts matrices

TABLE B-II
Description of Primary Overlays

PRIMARY OVERLAYS			
Overlay #	Program Name	Options Included	
(1,0)	OPT0	#0 - #9	
(2,0)	OPT10	#10 - #13, #15 - #17, #19	
(3,0)	OPT14U	#14 - Unknown Plants	
(4,0)	OPT14R	#14 - Regular Plants	
(5,0)	OPT14I	#14 - Irregular Plants	
(6,0)	OPT18	#18	
(7,0)	OPT20	#20 - #29	
(10,0)	OPTPLT	#30 - #39	
(11,0)	OPT31	#31 - #33	
(12,0)	OPT34	#34 - #36	
(13,0)	ERROR	All	
(14,0)	MEMORY	#99	
(15,0)	PRINT	#100 - #130	
Overlay #	Subroutines		Purpose
(7,0)	CLPASS		Form differential equations
	YOUT		Calculates output values

now organized and defined as shown in Table B-II.

2.3 MULTI's Data Elements

Labeled common blocks are used in MULTI to transfer data values between overlays. These data values are also retained in memory and determine the basic need for memory core space. Each COMMON block is selected so that only the data required by the primary overlay attached to the main overlay is transferred into the primary overlay. In addition, all character variables must be in separate blocks and not in numerical COMMON blocks.

The numerical sequence of the blocks corresponds to the overlay structure of the program. In COMMON block 7, the postscripts "A" and "S" refer to actuator and sensor. Table B-III shows which COMMON blocks are used in each overlay.

Arrays and matrix elements which are defined in the labeled common blocks are dimensioned in the same statements rather than with an additional set of DIMENSION statements. Arrays and matrices which are not common to more than one overlay are dimensioned in the individual overlay.

Initializations for the common blocks are accomplished via DATA statements in the executive overlay. All values are set equal to zero except as follows:

Actuators and sensors are set to be the transfer function...

$$\begin{tikzcd} {} \arrow[r, "e_\delta"] & \boxed{\dfrac{500}{s+500}} \arrow[r, "\delta"] & {} \end{tikzcd} \qquad (B-3)$$

Control limits are set equal to...

Minimum value —	-1.0×10^{10}
Maximum value —	1.0×10^{10}

TABLE B-III
COMMON Block Usage in MULTI's Overlays

	Overlay													
COMMON Block	(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)	(10,0)	(11,0)	(12,0)	(13,0)	(14,0)	(15,0)
B 1	X	X	X	X	X	X	X	X	X	X	X	X	X	X
B 2	X	X	X	X	X	X	X	X					X	X
B 3	X	X		X									X	X
B 4	X	X		X	X	X	X	X					X	X
B 5	X	X		X				X					X	X
B 6	X	X						X					X	X
B 7A	X	X						X					X	X
B 7S	X	X						X					X	X
B 8	X		X	X	X	X							X	X
B 8A	X		X	X	X	X		X					X	X
B 9	X		X	X	X	X		X					X	X
B10	X		X					X					X	X
B11	X		X			X	X	X					X	X
B12	X							X					X	
B12A	X							X					X	X
B12B	X							X	X				X	X
B13	X							X	X	X	X			X
B13A	X							X	X					X
B13B	X							X						X
B14	X								X	X	X			X
B14A	X								X	X	X			X
B15	X								X	X	X			X
B16	X								X	X	X			X
B17	X								X		X			X

X denoted that COMMON block is used in overlay

Initial values for all other arrays and matrices are entered in the individual overlays by the use of DO loops or DATA statements.

2.4 Program Labels

The program labels are selected to reflect certain operations in the program flow. Each label quickly identifies the code associated with the label and provides a method for easy editing and error checking. The MULTI program labels are defined as follows:

Label #	Purpose
0000 - 1999	Assigned sequentially in the program to label general operations.
2000 - 2999	Assigned to correspond to individual option transfers (i.e. OPTION #1 is labeled 2001).
3000 - 3999	Assigned to error statements in Overlay (13,0).
4000 - 4999	Assigned to FORMAT statements.
8000 - 8999	Assigned to error-free exit from overlays.
9000 - 9999	Assigned to CONTINUE statements for erroneous exit from overlays.

2.5 Overlay Structure

Each of MULTI's overlays are structured in the same basic format. The format has several elements beginning with overlay identification and ending with an END statement. Each overlay, however, may not contain all elements. The format used in MULTI's structure is as follows:

A. Overlay Beginning, including...
    - Overlay Identification
    - Program Name
    - CHARACTER, INTEGER, REAL Declarations
    - COMMON Blocks
    - DIMENSION Statements
    - DATA Statements

B. Option Flag Checking

C. Option Routing

D. Option/Overlay Code
    - Option Flag Initializations

E. FORMAT statements

F. Overlay Ending, including...
    - Error Flag Initializations
    - END Statement

Comment cards are used throughout the program to help a programmer interpret the overlays. In general, four lines of asterisks box an overlay-heading comment card, two rows of asterisks lead all subroutines and format blocks, and a single line of asterisks separates the options in each overlay. Finally, comment cards with dashed lines explain various operations or the next lines of code, or the dashed lines are used to separate sections of the program.

2.6 Overlay Calls

The main overlay is identified with a name and numerical designator:

```
OVERLAY (MULTI,0,0)
```

The primary overlays are identified only by numerical descriptors:

```
OVERLAY (n,0)
```

where "n" is an octal number.

Each overlay is called by designating the main overlay name, MULTI, and the numerical identification of the primary overlay with "n" in decimal. For example, to attach Overlay (11,0) the program call statement is:

```
CALL OVERLAY (MULTI,9,0)
```

This aspect of overlay usage is critical and is not defined well in the CDC FORTRAN V Reference Manual.

2.7 Option Flags and Error Flags

MULTI uses a simple, but effective, method to assure that options are accomplished in the correct order. This is necessary since program flow must begin with data input and proceed to simulation. As each option is accomplished, a corresponding option flag is set before program flow leaves the option.

To check program flow, the operational code beginning each overlay interrogates the option flags which must be set to meet the requirements of the option that is to be accomplished next. If all required options have not been accomplished, the related error flag is initialized. If an option is accomplished successfully, the option's error flag is set equal to zero.

Before each option request is signaled to the user, MULTI checks for error flags. If any error flags are set, the error overlay is automatically called to print an error message describing the problem with the program operation.

3. Description of MULTI's Main Executive Overlay

MULTI's main executive overlay controls the calling of the primary overlays and all flow of data between the overlays. All common memory locations are defined and initialized in the main executive overlay. Since the executive overlay is the most important overlay in the program, its operation should be thoroughly understood before the reader reviews the descriptions of all other overlays.

The following sub-sections describe how MULTI's executive, Overlay (0,0), is opened and calls primary overlays.

3.1 Overlay Opening

The main executive is opened with the overlay name and program statement. Type declarations for the variables occur next, followed by the COMMON block definitions and data initializations. The first two lines of operational code include a beginning message containing the current version number of the program.

The lines of code described above are executed only once; this occurs when the program is first begun. After MULTI is opened, the program flow remains in the option request-loop which begins at statement #9000.

3.2 Error Checking

All program-determined errors set an error flag denoting the type of error. When the program variable, IERR, is not equal to zero, Overlay (13,0) is called to warn the user of the error condition.

This error checking is the first operational line of the option request-loop code.

3.3 Option Request-Loop Code

The option request-loop begins with the statement:

```
PRINT '(//A)', ' OPTION, PLEASE> #'
```

and is used to obtain the option number request from the user and direct the program flow accordingly. The routing is accomplished by checking a series of IF/ELSEIF statements.

There are two additional checks accomplished when OPTION #14 is selected or if any plotting options are chosen.

If OPTION #14 is requested, the program asks the user to select the type of design that is to be accomplished. The single-character variable, METHOD, is used to further route the program flow to obtain the overlay needed for the computations.

When OPTIONS #30-39 are chosen, Overlay (10,0) is called to form a plotting matrix for terminal plots of CALCOMP plots. Upon completion of Overlay (10,0), the program returns to the main executive for more option checks. If a terminal plot is requested, Overlay (11,0) is called to accomplish the terminal plot. If OPTIONS #34 to 36 are requested to obtain a CALCOMP plot, Overlay (12,0) is brought into operational core to produce the PLOT file.

3.4 Main Executive Overlay Subroutines

There are four subroutines attached to the main overlay. These subroutines are used by more than one overlay and therefore remain in operational core at all times.

SUBROUTINE MATPR (TR,IR,IC) Subroutine MATPR is used for printing all matrices. The subroutine has three parameters as described below:

TR	- A real matrix of maximum dimension 10x10.
IR	- An integer value denoting the # of rows of the matrix.
IC	- An integer value denoting the # of columns of the matrix.

SUBROUTINE QPRINT (CHAR,*) This subroutine is used to ask if data should be printed. There are two parameters used in the subroutine call:

CHAR	- Character string with maximum of 30 characters.
*	- Line number denoting where program flow should go if data is not to be printed.

If the data is to be printed, the code directing the printing should follow immediately after the subroutine call.

SUBROUTINE ANSWER (*,*) ANSWER is used after the program echoes input data to the user and asks if the data is correct. If the answer from the user is affirmative, the program continues with the statements following the subroutine call. If the data is not correct or the option is aborted, the subroutine parameters direct the program flow as follows:

1st * - Line number denoting incorrect data directing return to data input point.
2nd * - Line number denoting option abort directing return to end of option.

For the second asterisk, the line number should be selected so as to set IERR equal to zero and so that the option flag is not set.

SUBROUTINE INVERT (A,AINV,N,IA,*) This subroutine is used to invert a matrix. The calling parameters are:

A	- A real matrix to be inverted of maximum dimension 10x10.
AINV	- The resulting real, inverted matrix.
N	- The # of rows in the A matrix.
IA	- The maximum row dimension of A matrix as described by the external program.
*	- Line number denoting where program flow should be directed if the A matrix cannot be inverted.

The line number parameter is selected to route program control to a point where an error flag is set corresponding to a program-directed error statement. This error statement indicates which matrix is not invertable.

Subroutine INVERT accomplishes the matrix inversion by use of the IMSL subroutine, LINV2F (Ref 14). The error flag from Subroutine LINV2F indicates if the inversion cannot be obtained. Print statements relating the problem are contained in Subroutine INVERT so that the user does not have to refer to the IMSL directives.

4. Description of MULTI's Primary Overlays

Program MULTI contains 13 primary overlays. The program code is written in FORTRAN V code and is easy to understand. Comments are included in some portions of the code to help explain the function of certain sections of the program.

Each overlay follows the structure described in Section 2.5. An additional structural aspect at the beginning of Overlays (1,0), (2,0), (7,0), (10,0), and (15,0) is the routing code to the individual options. Each overlay contains lines similar to the following line taken from Overlay (7,0):

```
GO TO (2021,2022,2023,2024,2025,2026,2027,2028,2029) NOPT
```

This line is used to route OPTIONS #20 to #29 to the lines beginning each option. NOPT is an integer variable which is equal to the option number the user requests. Upon entry into Overlay (7,0), NOPT's value is changed to NOPT-20. If NOPT then equals three, for example, the program flow is directed to label 2023 which is the third label listed in the GO TO statement. Option routing in the other overlays is similar. If NOPT's modified value becomes zero, program flow is to the statements directly following the GO TO statement. In case an error flag has been set during the overlay's operation, NOPT must be returned to its original value before leaving an overlay.

The reader can review the source listing for a description of each primary overlay. An overview of each of the overlays follows in this section. A listing of prerequisites for each option is found in Section 5.

4.1 Overlay (1,0) — Options #0 - #9

This overlay is used for entering data values which describe the plants. The code for the options in this overlay consists mainly of READ and PRINT statements.

OPTION #3 is used to enter the plant $\underline{A}$, $\underline{B}$, $\underline{C}$, and $\underline{D}$ matrices. It is noteworthy that the code is currently written so that if a matrix data entry is made incorrectly, the user must re-enter the entire matrix rather than change the individual element value. For this reason each matrix is echoed back to the user for checking by the calls to subroutines MATPR and ANSWER.

OPTIONS #4 and #5 include code to reset actuator and sensor values to the transfer function:

$$\begin{tikzcd} \arrow[r, "e_{\delta}"] & \boxed{\dfrac{500}{s + 500}} \arrow[r, "\delta"] & {} \end{tikzcd} \qquad \text{(B-4)}$$

so as to eliminate their effect in the simulation.

OPTIONS #8 and #9 read plant data from a local file. The data is read by opening a file, LFN, specified by the user. LFN is a variable that can be up to 30 characters in length. The file is first opened on a program-selected unit number, the data is read, and the unit is then closed. It is essential that the unit be closed at the end of the option so that multiple entries to the option can occur. This might be desired if the user wishes to introduce new plant data values for a different flight condition.

OPTIONS #6 and #7 are not used.

4.2 Overlay (2,0) — Options #10 - #19

This overlay contains the code for options #10 through #19, except that the code for OPTIONS #14 and #18 are contained in subsequent overlays. OPTIONS #15 thru #17 are not used.

The code for the options in this overlay is mostly composed of PRINT and READ statements pertaining to design parameters.

OPTION #19 contains the FORTRAN lines to read design parameters from a local file which must be specified by the user. The operation of data file opening and closing is given in the discussion of Overlay (1,0).

4.3 Overlay (3,0) — Option #14 - Unknown Plants

The controller matrices, $\underline{KO}$ and $\underline{K1}$, for unknown plants are calculated in Overlay (3,0). In addition to the COMMON blocks of the overlay, there are four additional scratch matrices (CM, VV, WW, and ZZ) dimensioned and initialized. The matrix values for these matrices are not retained after the overlay is terminated.

The unknown design can be accomplished by use of the plant $\underline{G}(0)$ matrix or by forming this matrix from the plant's $\underline{A}$, $\underline{B}$, $\underline{C}$, and $\underline{D}$ matrices. In the latter case the calculations are:

Program Calculation	Actual Calculation	
$VV = A^{-1}$	$VV = \underline{A}^{-1}$	(B-5)
$ZZ = VV * B$	$ZZ = [\ \underline{A}^{-1}\ \underline{B}\ ]$	(B-6)
$GO = D - C * ZZ$	$\underline{G}(0) = [\ \underline{D} - \underline{C}\ \underline{A}^{-1}\ \underline{B}\ ]$	(B-7)
$WW = D - C * ZZ$	$\underline{GM}(0) = \underline{G}(0)$	(B-8)

At this point either the calculated $\underline{G}(0)$ matrix or the user-originated $\underline{G}(0)$ matrix is used to create the controller matrices. The calculations are:

Program Calculation	Actual Calculation	
$WW = GO^{-1}$	$WW = \underline{G}(0)^{-1}$	(B-9)
$S = WW * GAMA$	$S = [\ \underline{G}(0)\ ]^{-1} \Sigma$	(B-10)
$K1 = S * EPSLON$	$\underline{K}_1 = \epsilon\ [\ \underline{G}(0)\ ]^{-1} \Sigma$	(B-11)
$KO = ALPHA * S * EPSLON$	$\underline{K}_o = \alpha\, \epsilon\ [\ \underline{G}(0)\ ]^{-1} \Sigma$	(B-12)

The single character variable, METHOD, is set at the end of the overlay to "U" if the overlay terminates normally; METHOD is set to "X" if the overlay terminates abnormally.

4.4 Overlay (4,0) — Option #14 - Regular Plants

This overlay calculates controller matrices, $\underline{K0}$ and $\underline{K1}$, for regular plants. There are three scratch matrices (VV, WW, and ZZ) which are dimensioned and initialized at the beginning of the overlay. The matrices are destroyed when the overlay is completed.

After prerequisite checks, the controller matrices are formed by the following calculations:

Program Calculation	Actual Calculation	
VV = C * B	$VV = [\, \underline{C}\, \underline{B} \,]$	(B-13)
ZZ = VV^{-1}	$ZZ = [\, \underline{C}\, \underline{B} \,]^{-1}$	(B-14)
WW = ZZ * GAMA	$WW = [\, \underline{C}\, \underline{B} \,]^{-1} \Sigma$	(B-15)
KO = WW * ALPHA * EPSILON	$\underline{K}_o = \alpha\, \epsilon\, [\, \underline{C}\, \underline{B} \,]^{-1} \Sigma$	(B-16)
K1 = WW * EPSILON	$\underline{K}_1 = \epsilon\, [\, \underline{C}\, \underline{B} \,]^{-1} \Sigma$	(B-17)

For a normal termination to the overlay, the single character variable, METHOD, is set equal to "R" denoting a regular plant design. If abnormal termination occurs, METHOD is set equal to "X".

4.5 Overlay (5,0) — Option #14 - Irregular Plants

An irregular plant design of the controller matrices, KO and K1, is accomplished in Overlay (5,0). There are four temporary matrices (VV, WW, ZZ, and VWZ) which are dimensioned and initialized for use only while this overlay is attached to operational core.

For this type of design the program requires a measurement matrix, MM, to augment the rank deficiencies of the output matrix. After prerequisite checks, the next block of program code determines if the measurement matrix already exists, if the user must initialize MM, or if the user desires to re-initialize the values of the matrix. The measurement matrix values are originally all set equal to zero in the main overlay. If the user has not initialized the values by selecting OPTIONS #9 or #18, the matrix values must be entered through this overlay.

The following calculations then occur:

Program Calculation	Actual Calculation	
WW = C + MM * A	$\underline{F} = \underline{C} + [\,\underline{M}\,\underline{A}\,]$	(B-18)
$ZZ = WW_2$	$\underline{F}_2 = [\underline{C}_2 + \underline{M}\,\underline{A}_{12}]$	(B-19)
$WW = ZZ^{-1}$	$WW = \underline{F}_2^{-1}$	(B-20)
ZZ = WW * GAMA	$ZZ = [\,\underline{F}_2\,]^{-1}\Sigma$	(B-21)
$WW = B_2$	$WW = \underline{B}_2$	(B-22)
$VWZ = WW^{-1}$	$VWZ = \underline{B}_2^{-1}$	(B-23)
WW = VWZ * ZZ	$WW = [\,\underline{F}_2\,\underline{B}_2\,]^{-1}\Sigma$	(B-24)

The controller matrices are then calculated as follows:

Program Calculation	Actual Calculation	
KO = WW * ALPHA * EPSLON	$\underline{K}_o = \alpha\,\epsilon\,[\,\underline{F}_2\,\underline{B}_2\,]^{-1}\Sigma$	(B-25)
K1 = WW * EPSLON	$\underline{K}_1 = \epsilon\,[\,\underline{F}_2\,\underline{B}_2\,]^{-1}\Sigma$	(B-26)

The overlay terminates by setting the single character variable, METHOD, equal to "I", denoting an irregular plant design, if the overlay terminates normally. It is set equal to "X" if the overlay terminates abnormally.

VMULFF, a high precision, IMSL Library subroutine (Ref 14), is used in this overlay to multiply two matrices together. When using this subroutine, the second matrix which is entered into the subroutine is overwritten during the matrix multiplication process. Thus, the contents of this matrix are destroyed.

4.6 Overlay (6,0) — Option #18

Overlay (6,0) is provided for two purposes. One purpose is to create a measurement matrix for use in an irregular plant controller design, and the other purpose is to provide a tool for row and column operations on any matrix.

This overlay uses the four scratch matrices (VV, WW, ZZ, and VWZ) during its computations. These matrices are discarded when the option ends.

The integer variable, ISKIP, is set equal to 1, 2, or 3 depending on the user's choice of measurement matrix creation, row and column operations on [$\underline{C}$ * $\underline{B}$], or row and column operations on any matrix of interest. The routing for the different choices is determined by an IF/ELSEIF block of code.

Before creating a measurement matrix, the user must obtain the feedback gain matrix, $\underline{K1}$, from CESA OPTION #38 (Ref 19:124). This $\underline{K1}$ matrix should not be confused with the $\underline{K1}$ matrix associated with the control law equations. After this matrix is read into the program as matrix VWZ, and is verified by the user, the measurement matrix is calculated by the following equations:

Program Calculations	Actual Calculations	
MM = C_1	MM = $\underline{C}_1$	(B-27)
WW = C_2	WW = $\underline{C}_2$	(B-28)
ZZ = MM - WW * VWZ	ZZ = $[\ \underline{C}_1 - \underline{C}_2\,\underline{K}_1\ ]$	(B-29)

<u>Program Calculations</u> (cont.)	<u>Actual Calculations</u> (cont.)	
$MM = A_{11}$	$MM = \underline{A}_{11}$	(B-30)
$WW = A_{12}$	$WW = \underline{A}_{12}$	(B-31)
$VV = WW * VWZ - MM$	$VV = [\ \underline{A}_{12}\, \underline{K1} - \underline{A}_{11}\ ]$	(B-32)
$WW = VV^{-1}$	$WW = [\ \underline{A}_{12}\, \underline{K1} - \underline{A}_{11}\ ]^{-1}$	(B-33)
$MM = ZZ * WW$	$MM = [\ \underline{C}_1 - \underline{C}_2\, \underline{K1}\ ]\ [\ \underline{A}_{12}\, \underline{K1} - \underline{A}_{11}\ ]^{-1}$	(B-34)

The program code for performing row and column operations on $[\ \underline{C}\ \underline{B}\ ]$ and performing row and column operations on any other matrix is identical. The only difference between the two choices is the matrix that is used in the operations. In the first case, the $[\ \underline{C}\ \underline{B}\ ]$ matrix is automatically formed by the program without any input from the user. In the second case, the user must supply a matrix of maximum dimension 10x10.

The program provides for four choices of operations on a matrix. The user's choice of operations, which is identified by the integer variable, ISKIP, directs program flow through an IF/ELSEIF block. When a new matrix is formed, it is echoed back to the user. The program then returns to the operation-selection point.

4.7 <u>Overlay (7,0)</u> — <u>Options #20 - #29</u>

All aspects of the controller simulation are contained in this overlay. The overlay also contains two subroutines, CLPASS and YOUT, which are used in OPTION #26. Since subroutine CLPASS is used as a parameter in a CALL statement, it must be declared as an EXTERNAL in the beginning of the overlay. There are two scratch matrices (CM and

F) and eight scratch arrays (IWORK, WORK, X, Z, Y, E, AX, and MMAX) that are dimensioned, initialized, used and discared as the overlay operates. The F matrix of this option should not be confused with the F matrix generated by OPTION #14 for irregular plants. There is also a CHARACTER variable called STRING which is declared at the beginning of the overlay. Overlay (7,0) contains one reserved option, OPTION #28.

Before the variable, NOPT, routes the program flow to the different options of the overlay, the CM and F matrix values must be set. Although these matrices are only superficial, it is necessary that they be equated to these values:

$$\underline{CM}\ (I,J) = \underline{C}\ (I,J) \tag{B-35}$$
$$\underline{F}\ (I,J) = \text{Identity Matrix} \tag{B-36}$$

These two matrices are found in the original program code obtained from the University of Salford and are used in OPTION #26.

OPTION #20 to #25 and OPTION #27 are mainly FORTRAN READ and PRINT statements which are used in obtaining state and integrator initial values, the output command vector, simulation time parameters and the control input limitations. OPTION #27 also contains the code to remove control limits by resetting the control minimum value to -1.0×10^{10} and the control maximum value to 1.0×10^{10}.

OPTION #29 opens unit number 40 to read the design parameters from a user-specified local file. The operation of this option is the same as the operation of OPTION #9 which is discussed in Section 4.1.

The heart of the simulation is contained in the OPTION #26 code. As previously stated, this code is nearly identical to the code received from the University of Salford. The option begins by checking for option flags which may not be set and by checking for the presence of actuators, sensors, and control limits. Next, the total number of states is determined by adding the number of states from the sensors, actuators and plant matrices.

There are three major loops in this option. The outer loop for each sampling time is blocked by label #1285; the middle loop based on sampling period is blocked by label #1250; the inner loop based on step time is blocked by label #1240.

Outer Loop. The outer loop occurs since the program allows the simulation to be run with more than one sampling time. Each time a different sampling time is run, the program must initialize several variable and array values. A calculation of the number of incremental points is also made, based upon the total simulation time, TT, the sampling time, SAMT, and the step time, ST. The following formulas determine the number of total time increments, NT:

$$\text{NIN} = \text{Integer Value of } (\text{TT} / \text{SAMT} + 0.5) \tag{B-37}$$
$$\text{NTT} = \text{Integer Value of } (\text{SAMT} / \text{ST} + 0.5) \tag{B-38}$$
$$\text{NT} = \text{NIN} * \text{NTT} \tag{B-39}$$

If the step time is chosen to be greater than the sampling time, NTT is set to a value of one prior to the calculation of total number of time increments in Eq. (B-39). The diagram in Fig. B-1 shows the relationship between TT, SAMT and ST.

If the total number of time increments is greater than 100, the simulation exceeds the dimension of the matrices which hold the input and output values. Thus, an integer variable, IPACK, is calculated to determine the rate at which data is packed into a matrix with 100 rows.

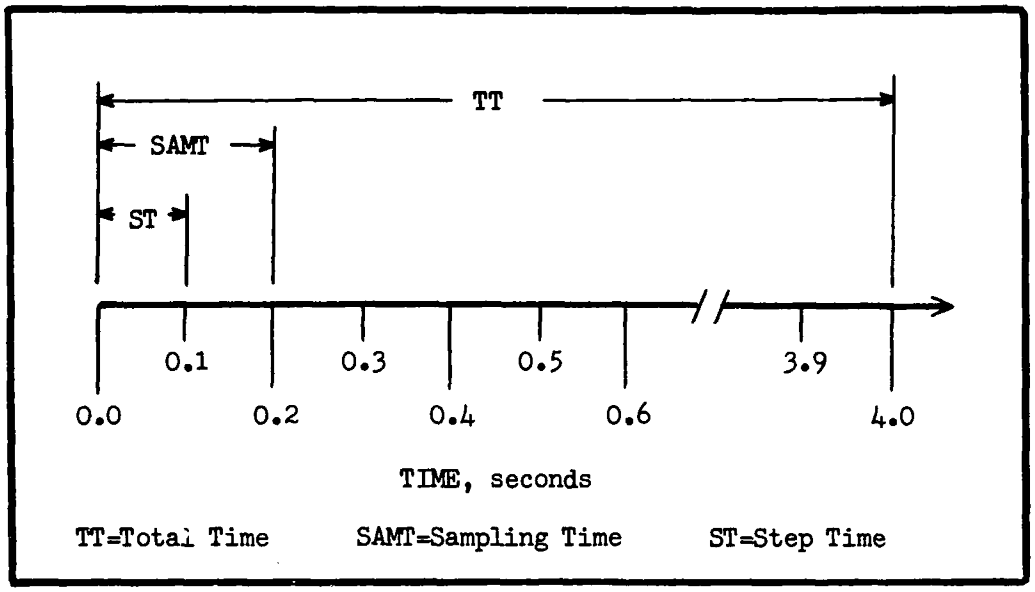


Fig. B-1 Relationship Between Total Time, Sampling Time and Step Time

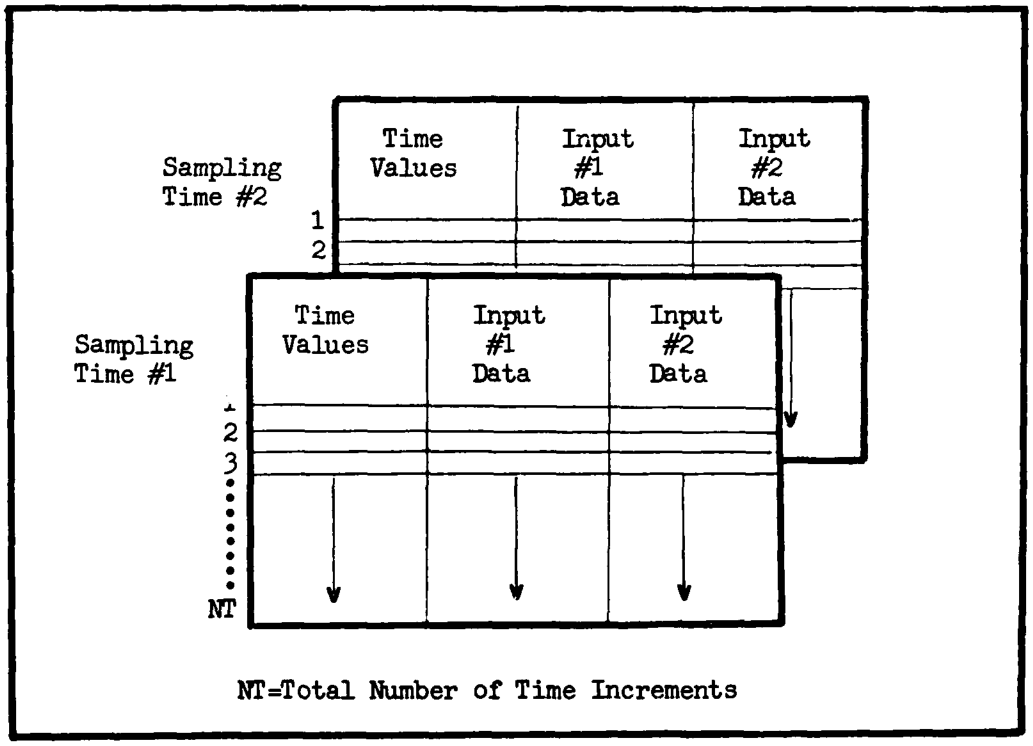


Fig. B-2 Representation of the 3-D Input Matrix, UP

If there are 125, or less, time increments, the data from the first 100 increments is retained. If there are more than 125 time increments, IPACK is calculated as:

$$\text{IPACK} = [\ \text{Integer Value of } (\ NT/100\ )\ ] + 1 \qquad \text{(B-40)}$$

As an example of the packing process, if IPACK is calculated as three, every third input and output value is retained.

The last part of the code for the outer loop sets the integer variable, ISKIP, which is used to suppress the time-sequential print-out of input and output values as the solutions to the state differential equations are found.

Middle Loop. The middle loop is passed through NIN times as calculated in Eq. (B-37). First, the output measurement vector, $\underline{YM}$, is formed. In MULTI this vector is equivalent to the output vector, $\underline{Y}$, and is calculated from:

$$\underline{Y} = \underline{CM} * \underline{X} \qquad \text{(B-41)}$$

The error vectors are then formed as follows:

for irregular plants...

Program Calculations	Actual Calculations	
$AX = A_1 * X$	$AX = [\ \underline{A}_1\ ]\ \underline{x}\ (kT)$	(B-42)
$MMAX = MM_1 * AX$	$MMAX = [\ \underline{M} * \underline{A}_1\ ]\ \underline{x}\ (kT)$	(B-43)
$S = F * Q$	$S = \underline{Q}$	(B-44)
$E = V - S - MMAX$	$\underline{E} = \underline{V} - \underline{Q} - [\ \underline{M} * \underline{A}_1\ ]\ \underline{x}\ (kT)$	(B-45)

for unknown plants and regular plants...

Program Calculations	Actual Calculations	
$S = F * Q$	$S = \underline{Q}$	(B-46)
$E = V - S$	$\underline{E} = \underline{V} - \underline{Q}$	(B-47)

For unknown plants the controller is then formed from:

$$\underline{U} = [\ \underline{KO} * \underline{E} + \underline{K1} * \underline{Z}\ ] \ / \ SAMT \qquad (B\text{-}48)$$

and for all other plants the controller is formed from:

$$\underline{U} = [\ \underline{KO} * \underline{E} = \underline{K1} * \underline{Z}\ ] \ SAMT \qquad (B\text{-}49)$$

However, before these inputs are used in the simulation they are compared to the current input control limitations and modified, if necessary.

A call to subroutine YOUT is the final line of code in the middle loop. This subroutine call determines the controlled output vector from the states of the system at any specified time and is described at the end of this section.

Inner Loop. The inner loop is passed through NTT times as calculated by Eq. (B-38). The first section of the loop is concerned with the data packing procedure described earlier and sequentially sets the row indices for the input and output matrices, UP and YP. These two matrices are retained throughout MULTI and contain the input and output data as calculated by OPTION #26. The matrices are 3-dimensional where the row dimension is equivalent to an input or output vector at each time increment, the first column holds the time increment values, T, while the other columns hold the sequential data for each input or output, and the third dimension is equivalent to the sampling time. See the diagram presented in Fig. B-2.

MULTI uses subroutine ODE (Ref 21) from the ASD CC6600 Library to solve the set of differential equations formed by subroutine CLPASS. The CLPASS subroutine is discussed at the end of this section. Currently the precision limi[illegible] [illegible]'s computations are set at 1.0×10^{-4}. If an error occurs during the solution process, an error message is printed to alert the user of the condition. The program provides the error flag number to the user and states that the ODE manual is to be referenced.

The inner loop is ended by a call to subroutine YOUT to form a new output vector, $\underline{Y}$, from the values returned from the ODE subroutine. When the inner loop is complete, a new integrator vector is calculated and program flow returns to the middle loop.

When all time increments for the total simulation time have been processed through the inner and middle loops, the program code for printing control limit information is entered. A simulation complete message is then provided and the outer loop is run again until each sampling time is simulated.

Before the option is completed the integer variable, NT, is set equal to the actual number of data points in the input and output matrices of the last simulation. This is necessary so that the plotting routines receive the correct number of data values in the matrices. However, since NT does not have a value for each sampling time and an input/output matrix can be formed where the first sampling time dimension has less data values than the second sampling time dimension, a termination error can result unless MULTI operates under one of the following procedures:

(a) MULTI should be run with all sampling times entered in order of increasing value, and step time should be set so as to be equal, or less then, the smallest sampling time, or
(b) MULTI should be run using only one sampling time.

SUBROUTINE CLPASS (T,X,XDOT)

This subroutine is nearly identical to the CLPASS subroutine attached to the PAK200 program. Minor modifications are made in some parts of the code so that it may be used as an external program in subroutine ODE.

This subroutine forms the differential equations from the plant, actuator, sensor, and error integral matrices. The parameters are:

T	Time
X	State vector
XDOT	Derivative of state vector, $\underline{X}$

CLPASS defines the values of the array, XDOT, at time, T. Array $\underline{X}$ includes all the states of the composite system. The plant states are first, followed by the actuator states, and then by the sensor states.

The system input, $\underline{U}$, is applied to the actuators. The output of the actuators, $\underline{W}$, provides the input to the plant. Finally, the plant output, $\underline{YM}$, is the input to the sensors, and the sensor output, $\underline{Q}$, is the actual system output. Fig. B-3 helps the reader visualize this arrangement.

SUBROUTINE YOUT (X,Y,W,C,D)

Subroutine YOUT is a modified version of the YOUT subroutine provided with PAK200. MULTI's YOUT subroutine uses a COMMON block to pass data into the subroutine rather than passing the data through the calling parameters.

This subroutine determines the controller output vector from the states of the system at any time. The parameters in the subroutine

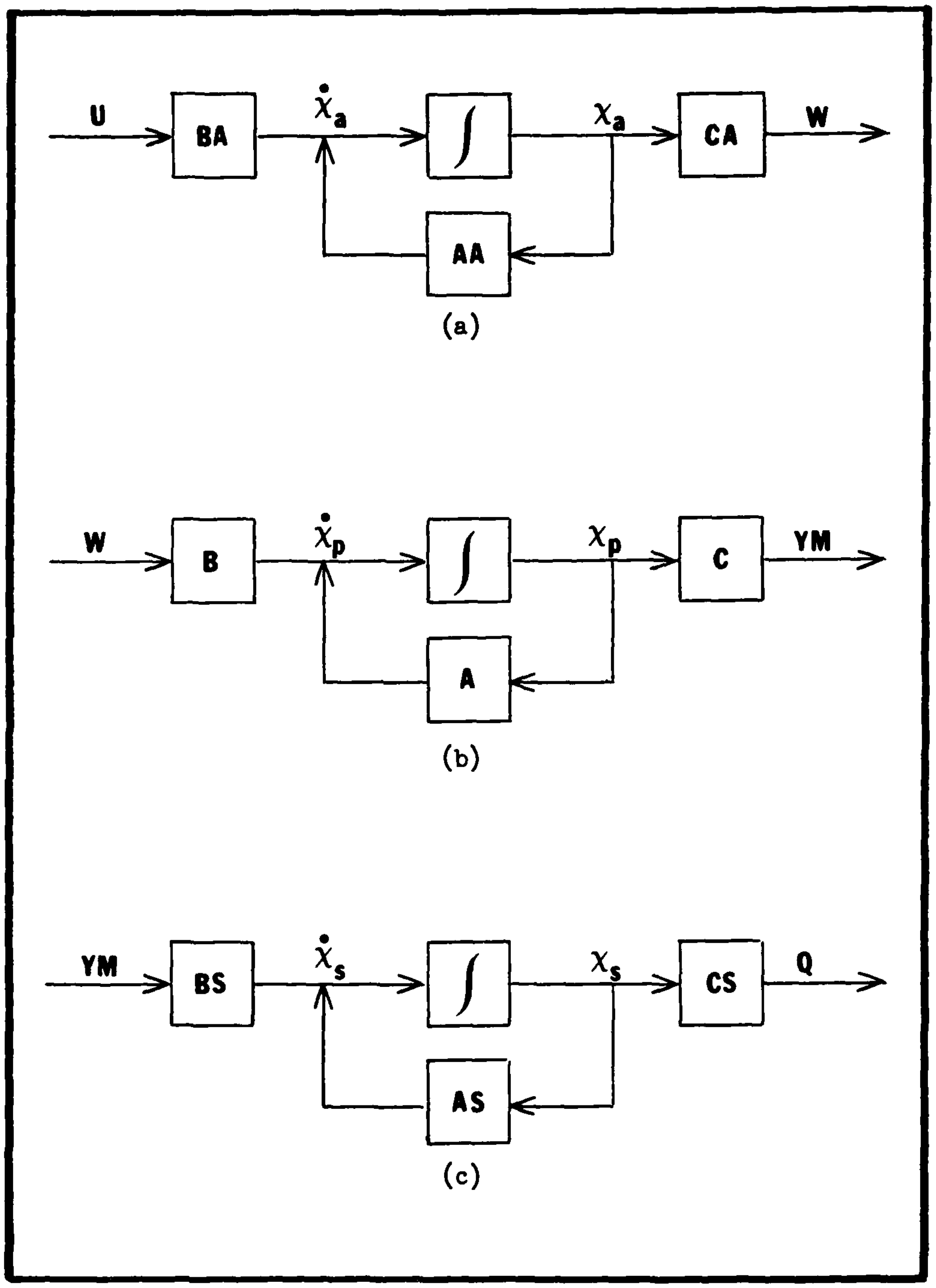


Fig. B-3 Block Diagrams for the
(a) actuators, (b) plant and (c) sensors

call are defined as:

X - System state vector
Y - Controlled output vector
W - Actuator output vector
C - System output matrix
D - System direct output matrix

4.8 Overlay (10,0) — Options #30 - #39

When any type of plot is to be generated, MULTI uses Overlay (10,0) to generate a plotting matrix, PLMAT, which is filled with data values from the input and/or output value matrices, $\underline{UP}$ and $\underline{YP}$. After the PLMAT is formed the actual terminal plot is generated by Overlay (11,0) while the actual CALCOMP plot is generated by Overlay (12,0). These two overlays are discussed in the next two sections.

It is pointed out to the reader that COMMON blocks B13 and B13A change the character representations of the integer variable, NT, and the matrices, $\underline{YP}$ and $\underline{UP}$. In this overlay and Overlays (11,0) and (12,0), these variable names are referred to as N, Y, and P, respectively.

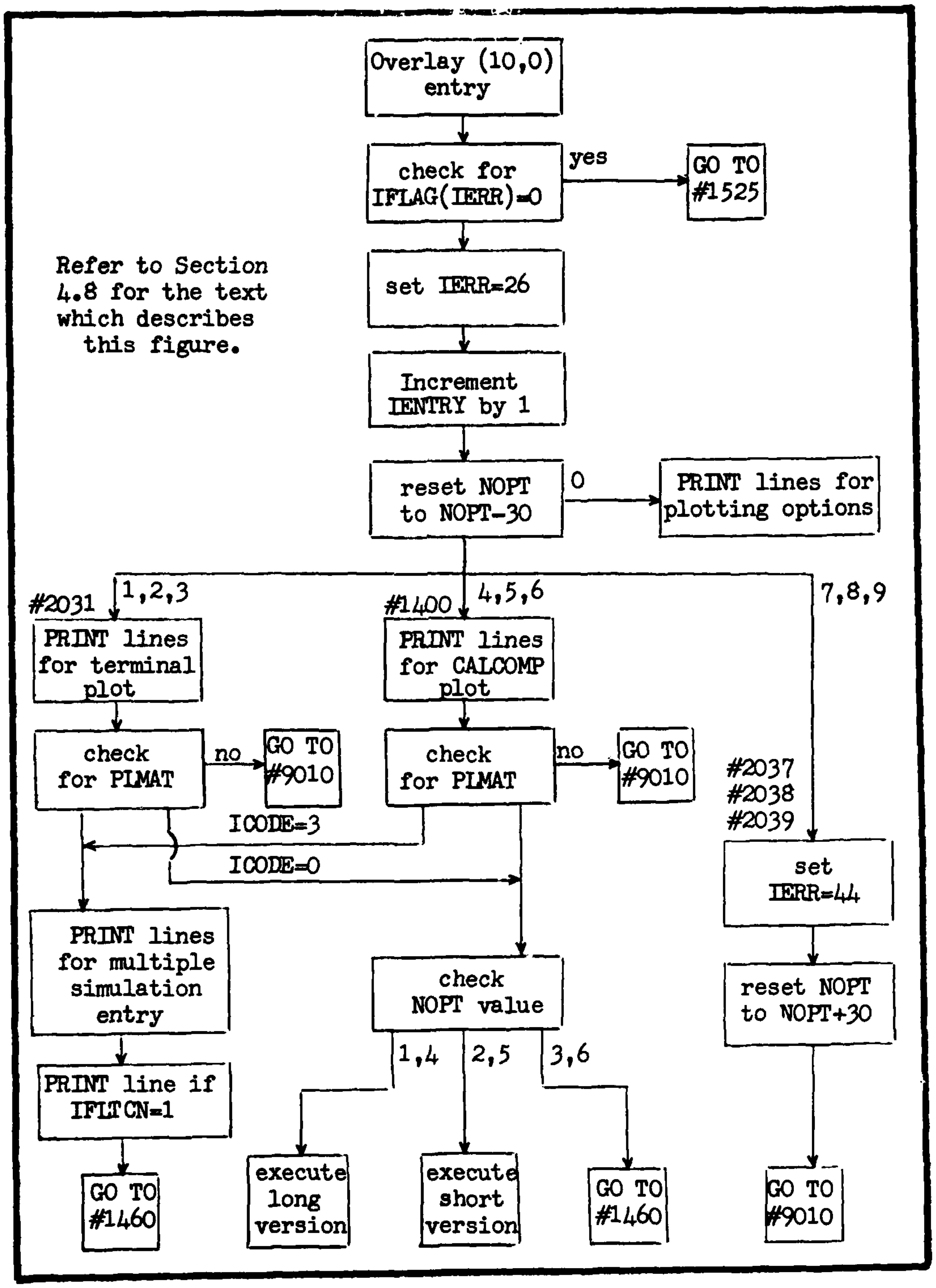
The FORTRAN coding in this overlay is very condensed and the program flow is very complex. The coding is condensed so that the overlay can operate successfully while using minimum core space; the core space requirement is critical since this overlay uses the three largest matrices of the program ($\underline{YP}$, $\underline{UP}$, and PLMAT). The program flow is complex since MULTI provides (1) terminal plots and CALCOMP plots, (2) four types of each kind of plot, and (3) a long version and two shortened versions of the plot request. All of these provisions are completed in this single overlay. In addition, if PLMAT is generated for several different flight conditions, the program flow loops between this overlay and the other options of the program until PLMAT is completed. In this case, the overlay is required to keep a count of the number of times that the plot sequence has been entered. Figure B-4 shows the program routing at the beginning of the overlay. In the figure, the single digit numbers, beside the lines connecting the blocks, correspond to the value of NOPT. The four digit numbers above the blocks correspond to actual program label numbers.

To correctly complete a plotting matrix, PLMAT, this overlay extracts columns of data from the U and Y matrices. The columns are transferred to PLMAT in a predesignated order after the user inters the values for the variables ICODE, NPAIRS, IFLTCN, LINES, NDEPTH, IDEPTH, CHOICE, NUM, ICOLMN, and ICLM. The program flow used to set these variables is shown in Fig. B-5. The values for these variables may be set by the program, rather than by the user, depending on the type of plot requested. The definitions for these variables are found in the last section of this manual. Figure B-6 shows the composition of the different PLMAT matrices which are formed for the four types of plots.

After the PLMAT matrix is complete, the integer variables, NUM and ICLM, are reset to the values required for correct operation in Overlay (11,0) and (12,0). This ends the overlay.

4.9 Overlay (11,0) — Terminal Plot

MULTI has the capability to produce terminal plots by calling upon subroutine PLOTIT. PLOTIT is a subroutine designed by Major



Refer to Section 4.8 for the text which describes this figure.

Fig. B-4 Initial Program Routing for Overlay (10,0)

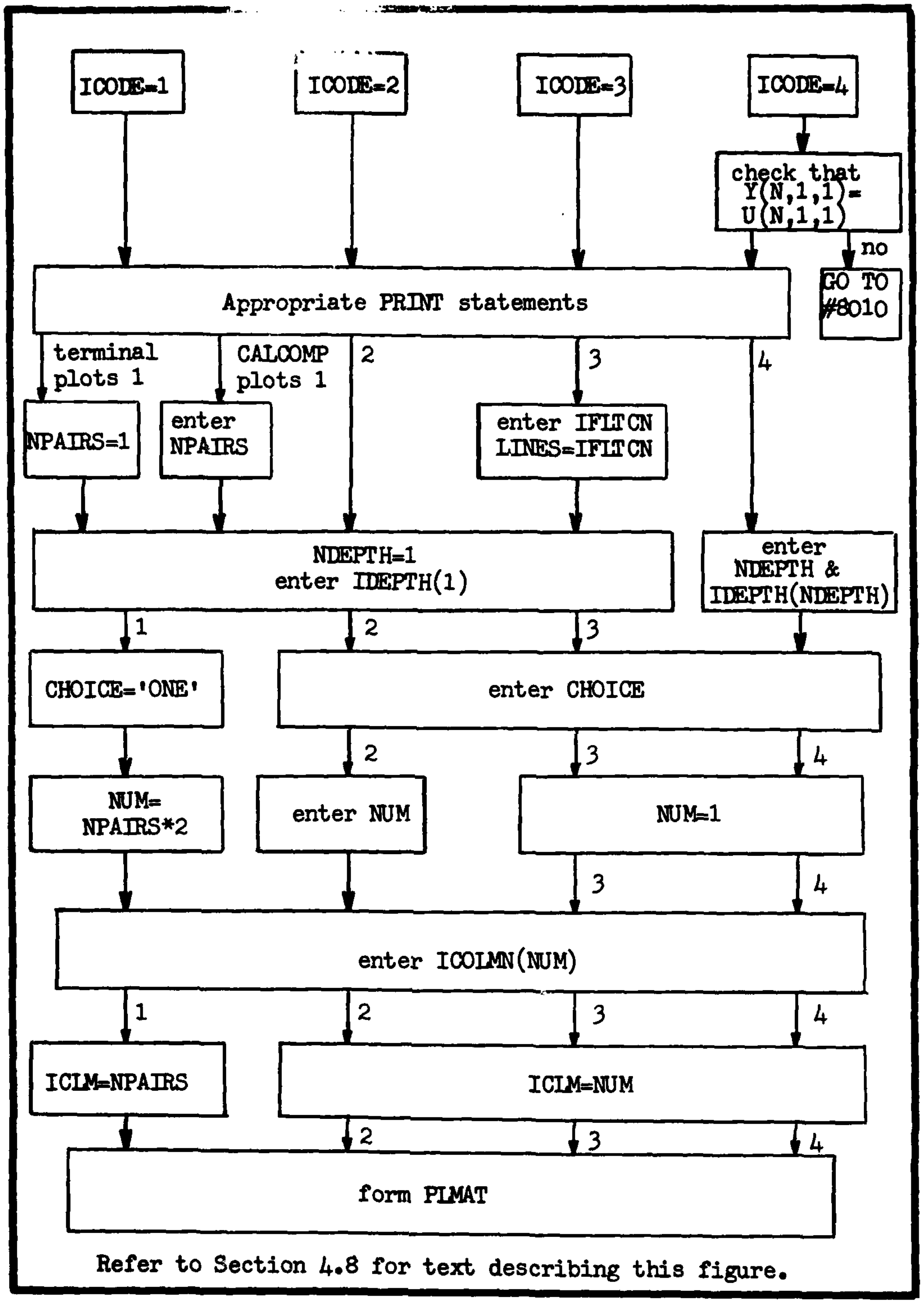


Fig. B-5 Program Routing Used to Set Variables Needed to Form the Plotting Matrix, PLMAT

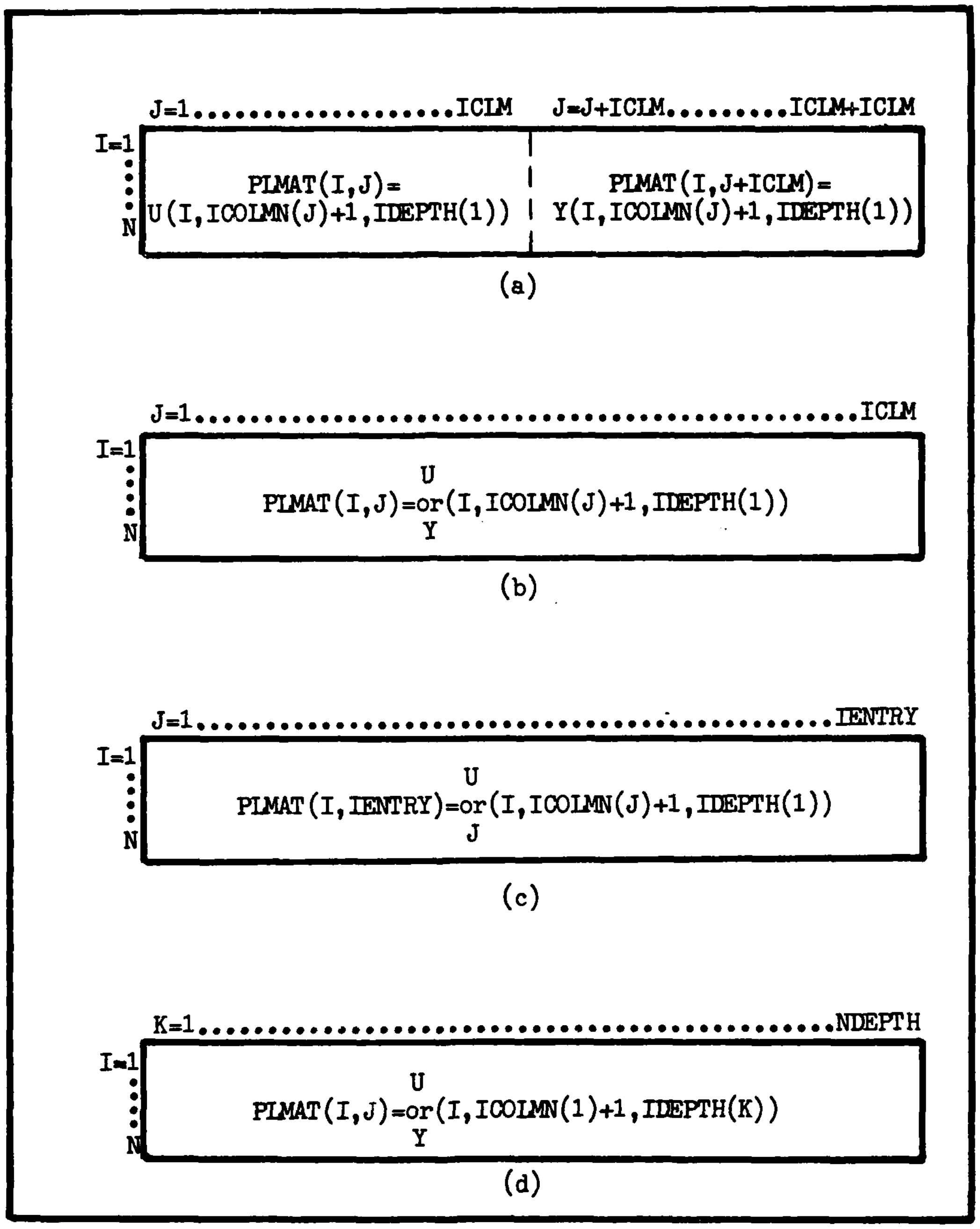

J=1.....................ICLM
J=J+ICLM.........ICLM+ICLM
I=1
N
PLMAT(I,J)=
U(I,ICOLMN(J)+1,IDEPTH(1))
PLMAT(I,J+ICLM)=
Y(I,ICOLMN(J)+1,IDEPTH(1))
(a)
J=1..ICLM
I=1
N
U
PLMAT(I,J)=or(I,ICOLMN(J)+1,IDEPTH(1))
Y
(b)
J=1..IENTRY
I=1
N
U
PLMAT(I,IENTRY)=or(I,ICOLMN(J)+1,IDEPTH(1))
J
(c)
K=1..NDEPTH
I=1
N
U
PLMAT(I,J)=or(I,ICOLMN(1)+1,IDEPTH(K))
Y
(d)

Fig. B-6 Composition of PLMAT Matrix for (a) ICODE=1, (b) ICODE=2, (c) ICODE=3, (d) ICODE=4

Michael R. Stamm of the Department of Physics at the Air Force Institute of Technology, Wright-Patterson AFB, Ohio (Ref 23).

This overlay begins with the addition of a new COMMON block called SCALIT which is common only to the PLOTIT subroutine. The COMMON block provides for the initialization of the terminal plot's calling parameters. As discussed in Section 4.8, this overlay also renames the variables, NT, $\underline{UP}$, and $\underline{YP}$ as N, $\underline{U}$, and $\underline{Y}$, respectively.

Overlay (11,0) must also provide the PLOTIT subroutine with the independent and dependent data arrays, XAXIS and YAXIS, plus the minimum and maximum values of these arrays. The XAXIS array is always filled with values from the first column of the output matrix, $\underline{Y}$. The YAXIS is sequentially set equivalent to the columns of the PLMAT matrix as the PLOTIT subroutine is called. PLOTIT is called until all the data in the PLMAT is scanned and entered into the terminal plot routine. The terminal plot is then generated.

The overlay is ended with a series of PRINT statements which relate the symbols on the terminal plot curves to the input/output numbers of the plant states.

4.10 Overlay (12,0) — CALCOMP Plot

This section describes the code in Overlay (12,0) which uses the data in the PLMAT matrix and produces a CALCOMP plot. The description of the subroutine calls in the ASD CALCOMP Plotter Guide (Ref 22) is minimal, at best. This fact, combined with the fact that the AFIT computer terminal produces CALCOMP plots at a glacial pace, makes CALCOMP plotting code difficult to write and test. Since help with this type of code is limited, for AFIT students, the discussion of the

CALCOMP subroutines included in this section has the purpose of augmenting the information in the CALCOMP plotting guide.

The plot, which is produced by the code of this overlay, has an X-axis length of eight inches and a Y-axis length of five inches. Each axis is labeled with a title and scaling values. The independent axis is automatically titled: TIME, SECONDS. All curves are computer approximations of a smooth line between the discrete data points. A number is placed at the end of each curve to identify the curve as representing a specific input/output. Each plot is finished with a rectangular box with outside dimensions of 6x9 inches. The plot's title is drawn outside the box and below the lower nine inch side.

The overlay renames the variables, N, $\underline{UP}$, and $\underline{YP}$, to N, $\underline{U}$, and $\underline{Y}$, respectively, as discussed in Section 4.8. In addition to the variables, arrays, and matrices included in the COMMON blocks, Overlay (12,0) creates four arrays called XAXIS, YAXIS, ABSC, and ORD, and the CHARACTER variable, XTITLE. The four arrays are initialized to zero values while the CHARACTER variable is initialized to produce the X-axis title.

The first executable line of the overlay records the number of CALCOMP plots that are generated by incrementing the integer, IPLOT. When the file, PLOT, contains the data for five plots, the program prompts the user to terminate MULTI and route the PLOT file to the printer.

The array of independent axis data points, XAXIS, is formed by copying the first column of data from the output data matrix, $\underline{Y}$. The same XAXIS data array is used when plotting all curves. The program next finds the minimum and maximum values that are contained in the PLMAT matrix and forms an artificial YAXIS data array containing these values. This dummy YAXIS array is used only to scale the dependant axis and is overwritten when the actual plots are formed.

The next portion of the code deals with obtaining the titles for the Y-axis and the plot. The former is contained in the 30-letter CHARACTER variable, TITLE, and the latter is contained in the 60-letter CHARACTER variable, LFN. The titles should be centered between the ">" prompts, if the titles are to be centered on the CALCOMP plot. The program calls subroutine ANSWER to verify each title.

At this point the CALCOMP plotting calls begin. The two ASD Libraries, CCPLOT56X and CCAUX, hold all the plotting subroutines needed for MULTI's CALCOMP plotting capability. The plotting calls begin with the mandatory call to PLOTS which allows access to all other plotting subroutines. Subroutine FACTOR is then used to set the size of the plot. For a unity scaling factor, the independent axis values are scaled across an eight inch axis and the dependent axis values are scaled across a five inch axis by the subroutine SCALE. The adjusted minimum and adjusted delta, computed by this subroutine, is placed in the (N+1) and (N+2) rows, respectively, of all PLMAT matrix columns. This is accomplished so that the columns of data can all be placed on the same graph. These adjusted values are also entered into the last two rows of the ABSC and ORD arrays.

The plot's origin is then established at a point which is two inches above and two inches right of the CALCOMP reference position.

Actual drawing begins with two calls to subroutine AXIS; this draws and labels the X and Y axes.

If a plot is produced with negative Y-axis values, subroutine LINE is called to draw a line dividing the positive and negative portions of the plot. The line extends horizontally from the coordinate origin, (0,0). Since the coordinate origin may not be included as a data point in PLMAT, the ABSC and ORD arrays are generated. The ABSC and ORD arrays contain two data points each. Together they form the coordinates (0,0) and (0,XAXIX(N)). The third and fourth element of each array holds, respectively, the plot's adjusted minimum and adjusted delta. Subroutine LINE is used to draw a straight line between the two points, thus creating an unlabeled X-axis.

The plotting loop begins at statement #1720. To plot each curve, the YAXIS array is sequentially set equal to each column of the PLMAT matrix. The coordinates, where the symbol at the end of each curve is placed, are calculated and defined as (XSYM,YSYM). This occurs each time the YAXIS array is defined. Subroutine FLINE is used to plot each curve. This subroutine connects all data points with a smooth line and places the appropriate symbol at the end of each curve by referencing the STRING variable called STRING.

When the plotting loop ends, the plot is boxed by calling subroutine RECT. The plot title is drawn by a call to SYMBOL. The plotting calls are then terminated by the mandatory call to PLOTE.

The overlay ends with two groups of code. The first group contains printing statements, and the second group reinitializes the variables, IERR, IENTRY, and ICODE, to zero.

The final part of this section discusses aspects about several ASD Library plotting subroutines, which are not stressed in Reference 22.

Subroutine PLOTS (A,B,M) The parameter names, used when calling this subroutine, must reflect the correct type (i.e. REAL, INTEGER) of variable. Although the parameters are ignored and the variables can have any name, the names must not be used in other portions of the program.

Subroutine FACTOR (FACTR) If the parameter, FACTR, is set to a unity value, plots sent to AFIT (terminal #91) have an outer box of size 6x9. Plots sent to AFWAL (terminal #90) are automatically reduced by a factor of 0.5; thus, to produce a plot of program-designed size, FACTR must be set to a value of two. The automatic features of other base terminals are unknown to the author.

Subroutine PLOT (X,Y,IC) At least one call to this subroutine must be made to define a plotting origin. If a call to PLOT is not made, the X and Y coordinate origin can be considered as the CALCOMP pen's resting position. Therefore, the plotting origin must be established so that the plot is assured of being drawn on the paper.

Subroutine AXIS (X,Y,BCD,NC,SIZE,THETA,AMIN,DA) In the description of this subroutine, the text states that the axis title is placed counterclockwise or clockwise in relation to the axis. The user should consider this definition to mean above or below the axis.

Subroutine SYMBOL (X,Y,HGHT,IBCD,THETA,N) The X and Y parameters in this subroutine are defined in terms of inches from the plotting origin as defined by the call to subroutine PLOT, not from the coordinate origin of the plot. To obtain coordinate values in inches to use in this subroutine, the user must reference the discussion of subroutine OFFSET. As an example, the X-coordinate, XSYM, for the symbol at the end of each curve is calculated as:

$$XSYM = \frac{XAXIS(N) - \text{adjusted minimum}}{\text{adjusted delta}} \qquad \text{(B-50)}$$

The adjusted minimum and adjusted delta are calculated in the call to subroutine SCALE and the program places these values in the (N+1) and (N+2) rows of the PLMAT matrix. The YSYM coordinate is calculated in the same manner.

To obtain an upright, or vertical, symbol, the value for the THETA parameter is set equal to zero.

Subroutine FLINE (X,Y,N,K,J,L) The J parameter in this subroutine is correctly defined as:

$J < 0$	Draw a symbol at every Jth point.
$J > 0$	Connect all points with a straight line or a smooth curve and draw a symbol at every Jth point.

4.11 Overlay (13,0) — Error Statements

This overlay is used to print error messages for all operational failures detected by the program. The operational code of the overlay consists entirely of PRINT statements.

The first 30 error statements refer to option errors. The number of the error statement relates directly to the number of an error flag which, in turn, corresponds directly to an option number. Error statements #31 thru #39 provide information about matrix errors. Error statements #40 thru #44 give miscellaneous error information. When an option does not exist, a PRINT statement is still included to provide an operational check of MULTI. If the user receives one of these error statements, there is a program flow problem which should be checked by a programmer.

As a working example, consider a prerequisite check at the beginning of an option. OPTION #3 requires the option flag from OPTION #2 to be set before the plant matrices can be entered, If IFLAG(2) is not set prior to selecting OPTION #3, IERR is set equal to two and Overlay (13,0) is called. A value of two for IERR routes the program flow to statement #3002 which prints:

```
# OF STATES, INPUTS & OUTPUTS MISSING...SEE OPTION #2
```

4.12 Overlay (14,0) — Option #99 - Memory Files

When the user selects OPTION #99 to end program MULTI, Overlay (14,0) is called to create the data memory files. There are three memory files created.

MEMO is created to hold plant data as directed by the integer variable, IPLANT. If IPLANT equals "1", all state, sensor and actuator matrices are saved; if IPLANT equals "2", the plant $\underline{G}(0)$ matrix is copied to the file. If IPLANT equals "0", plant data from OPTIONS #2 to #5 are still transferred to the memory file, and, when the file is read back into MULTI, the IPLANT value alerts the user that the file's data may be incomplete.

MEM10 is used to save all design data, and MEM20 is opened to contain the simulation data.

The overlay ends with several PRINT statements relating the file names that have been created and the data that each contains.

4.13 Overlay (15,0) — Options #100 - #139

MULTI's last overlay provides data value printing for most of the options. If an option is not concerned with data input or data creation, the program uses error flag #43 to print:

THERE ARE NO VALUES IN MEMORY CORE FOR OPTION #

If data is available, the values are printed by routing the program flow to the proper section of the overlay. All printing options are numbered by adding the value of 100 to the related option number of the main program. For example, data values entered or created by OPTION #12 are printed by OPTION #112.

The data values from some options are combined to form a single block of data information. The data from OPTIONS #2 and #3 is printed when using OPTION #103. The data from OPTIONS #11 and #13 is generated when either OPTION #111 or #113 is selected. The same occurs with data from OPTIONS #21 and #22; the data is printed by using either OPTION #121 or #122. And finally, all data values from OPTIONS #23 to #25 are provided by selecting any option number from 123 to 125.

The code for the overlay is simple and self-explanatory. The reader is directed to the MULTI source listing for further detail on this overlay's operation.

5. Option Pre-requisites

The following list provides the prerequisite requirements for MULTI's options.

OPTION #	PRE-REQUISITE OPTION #
3,4,5,12,21,22,29	2
14	2, 3, 11 to 13
18 (entries 1 & 2)	5
26	14, 21 to 25
31,32,34,35	26
33	31 or 32
36	34 or 25

6. Variable Names and Definitions

There are some variables in the program whose purpose is not easily identified or described by their program variable name. The following list provides further information for these variables.

CHARACTER Variables	Purpose/Definition
ACT	Flag for actuators
DMATRX	Flag for D matrix
SEN	Flag for sensors
PLTYPE	Matrix printing indicator in Overlay (15,0)

INTEGER Variables	Purpose/Definition
ICLM	Running variable of ICOLMN
ICODE	Denotes type of plot being requested
ICOLMN	Vector of specific input/output numbers for plots
IDEPTH	Sampling time of interest for plots
IFLTCN	Number of simulations being placed on one plot
IPLOT	Number of plots on PLOT file
M	Number of states of plant
MD	Maximum number of states
N	Number of inputs of plant
NA	Number of actuators
ND	Maximum number of inputs
NDEPTH	Running variable of IDEPTH
NPAIRS	Number of input/output pairs to be plotted
NS	Number of sensors
NUM	Number of columns of data in PLMAT
P	Number of outputs of plant
PD	Maximum number of outputs

REAL Variables	Purpose/Definition
GN	Vector of all sampling times
PLMAT	Plotting matrix
UP	Input data matrix
YP	Output data matrix

Appendix C: MULTI Source Listing

The following code is the FORTRAN V MULTI source listing:

```
C **********************************************************************
C **********************************************************************
C  EXECUTIVE -- MULTI
C **********************************************************************
C **********************************************************************
      OVERLAY (MULTI,0,0)
      PROGRAM EXEC
      CHARACTER MULTI*10,DMATRX,METHOD,CHOICE*6,ACT,SEN
      INTEGER P,PD
      REAL K0,K1,MM
      COMMON /B  1 / NOPT,IERR,IPLANT,IFLAG(40)
      COMMON /B  2 / N,M,P,ND,MD,PD
      COMMON /B  3 / G0(10,10)
      COMMON /B  4 / A(10,10),B(10,10),C(10,10)
      COMMON /B  5 / D(10,10)
      COMMON /B  6 / DMATRX,ACT,SEN
      COMMON /B  7A/ NA(10),AA(10,2,2),BA(10,2),CA(10,2)
      COMMON /B  7S/ NS(10),AS(10,2,2),BS(10,2),CS(10,2)
      COMMON /B  8 / ALPHA,EPSLON,GAMA(10,10)
      COMMON /B  8A/ METHOD
      COMMON /B  9 / K0(10,10),K1(10,10)
      COMMON /B 10 / NG,GN(10)
      COMMON /B 11 / MM(10,10)
      COMMON /B 12 / W(10),Q(10),U(10),YM(10)
      COMMON /B 12A/ X0(10),Z0(10),V(10)
      COMMON /B 12B/ ST,TT
      COMMON /B 13 / NT,YP(100,11,2)
      COMMON /B 13A/ UP(100,11,2)
      COMMON /B 13B/ ILIM,IMIN(10),IMAX(10),UMIN(10),UMAX(10)
      COMMON /B 14 / ICODE,NUM,ICOLMN(4),PLMAT(102,4)
      COMMON /B 14A/ CHOICE
      COMMON /B 15 / ICLM,NPAIRS,NDEPTH,IDEPTH(2)
      COMMON /B 16 / IENTRY,IFLTCN,LINES
      COMMON /B 17 / IPLOT
      DATA MULTI,ACT,SEN/'MULTI','N','N'/
      DATA NOPT,IERR,IPLANT,N,M,P/6*0/
      DATA ND,MD,PD/3*10/
      DATA ALPHA,EPSLON,ST,TT,NG,NT,ICODE,NUM/8*0/
      DATA ICLM,NPAIRS,NDEPTH,IENTRY,IFLTCN,LINES,IPLOT/7*0/
      DATA GN,X0,Z0,V/40*0./
      DATA NA,NS,AA,AS,BA,BS,CA,CS/20*1,80*-500.,40*500.,40*1./
      DATA IFLAG,W,Q,U,YM/40*0,40*0./
      DATA G0,A,B,C,D,GAMA,K0,K1,MM/900*0./
      DATA ILIM,IMIN,IMAX,UMIN,UMAX/21*0,10*-1.E10,10*1.E10/
      DATA PLMAT,YP,UP/4808*0./
C --------------------BEGINNING MESSAGE---------------------------------
      PRINT '(//A)', ' WELCOME TO MULTIVARIABLE DESIGN'
      PRINT '(A)', ' C 1981    PORTER,SMYTH,PASCHALL'
      PRINT*, ' '
      PRINT*, 'THIS PROGRAM USES THE DESIGN TECHNIQUES DEVELOPED BY'
      PRINT*, 'PROFESSOR BRIAN PORTER, UNIV. OF SALFORD, ENGLAND'
C --------------------OPTION CALL---------------------------------------
 9000 IF (IERR.NE.0) CALL OVERLAY (MULTI,11,0)
      PRINT '(//A)', ' OPTION, PLEASE > #'
      READ*, NOPT
      PRINT*, ' '
C --------------------ROUTING FOR OPTIONS #0 TO #9---------------------
      IF (NOPT.LT.10) THEN
          CALL OVERLAY (MULTI,1,0)
          GO TO 9000
C --------------------ROUTING FOR OPTIONS #10 TO #19-------------------
      ELSEIF (NOPT.EQ.14) THEN
    3     PRINT*, 'ENTER DESIGN METHOD...UNKNOWN,REGULAR,IRREGULAR...>'
          READ '(A)', METHOD
      PRINT*, ' '
          IF (METHOD.EQ.'U') THEN
              CALL OVERLAY (MULTI,3,0)
              GO TO 9000
          ELSEIF (METHOD.EQ.'R') THEN
              CALL OVERLAY (MULTI,4,0)
              GO TO 9000
          ELSEIF (METHOD.EQ.'I') THEN
              CALL OVERLAY (MULTI,5,0)
              GO TO 9000
          ELSE
              GO TO 3
          ENDIF
      ELSEIF (NOPT.EQ.18) THEN
          CALL OVERLAY (MULTI,6,0)
          GO TO 9000
      ELSEIF (NOPT.GE.10.AND.NOPT.LT.20) THEN
          CALL OVERLAY (MULTI,2,0)
          GO TO 9000
C --------------------ROUTING FOR OPTIONS #20 TO #29-------------------
      ELSEIF (NOPT.GE.20.AND.NOPT.LT.30) THEN
          CALL OVERLAY (MULTI,7,0)
          GO TO 9000
C --------------------ROUTING FOR OPTIONS #30 TO #39-------------------
      ELSEIF (NOPT.GE.30.AND.NOPT.LE.39) THEN
          CALL OVERLAY (MULTI,8,0)
          IF (IERR.NE.0) GO TO 9000
          IF (NOPT.EQ.0) GO TO 9000
          IF (IFLTCN.NE.0) GO TO 9000
C --------------------ROUTING FOR SPECIFIC TYPE OF PLOT----------------
          GO TO (5,5,5,10,10,10,14,14,14) NOPT
  5         CALL OVERLAY (MULTI,9,0)
            GO TO 9000
 10         CALL OVERLAY (MULTI,10,0)
            GO TO 9000
 14         GO TO 9000
C --------------------ROUTING FOR OPTION #99--------------------------
      ELSEIF (NOPT.EQ.99) THEN
          CALL OVERLAY (MULTI,12,0)
          PRINT '(/A/)', ' HAVE A NICE DAY!'
          STOP
C --------------------ROUTING FOR OPTIONS #100 TO #130-----------------
      ELSEIF (NOPT.GE.100.AND.NOPT.LE.130) THEN
          CALL OVERLAY (MULTI,13,0)
          GO TO 9000
C --------------------ROUTING FOR NON-AVAILABLE OPTIONS----------------
      ELSE
          IERR=44
          GO TO 9000
      ENDIF
C -------------------------------------------------------------------
      END
C ******************************************************************
C ******************************************************************
C THIS SUBROUTINE IS USED TO PRINT ALL MATRICES
      SUBROUTINE MATPR (TR,IR,IC)
      DIMENSION TR(10,10)
      DO 15 I=1,IR
 15   PRINT 4005, (TR(I,J),J=1,IC)
      PRINT*, '  '
4005  FORMAT (3X,10(1E10.4,3X))
      END
C ******************************************************************
C ******************************************************************
C THIS SUBROUTINE IS USED TO ASK IS DATA SHOULD BE PRINTED
      SUBROUTINE QPRINT (CHAR,*)
      CHARACTER CHAR*30
      PRINT '(/A,A)', ' ENTER "0" TO SKIP ',CHAR
      PRINT*, 'ENTER "1" TO OBTAIN THIS DATA PRINTOUT...>'
      READ*, IPRINT
      IF (IPRINT.NE.1) RETURN 1
      END
C ******************************************************************
C ******************************************************************
C THIS SUBROUTINE IS USED TO ASK IF DATA IS CORRECT
      SUBROUTINE ANSWER(*,*)
      CHARACTER ANSER
      PRINT '(/A)' , ' IS THIS CORRECT...YES,NO,$...>'
      READ '(A)', ANSER
      PRINT*, '  '
      IF (ANSER.EQ.'Y') GO TO 20
      IF (ANSER.EQ.'N') RETURN 1
      IF (ANSER.EQ.'$') RETURN 2
   20 CONTINUE
      END
C **********************************************************************
C **********************************************************************
C THIS SUBROUTINE IS USED TO INVERT ALL MATRICES
      SUBROUTINE INVERT(A,AINV,N,IA,*)
      DIMENSION WKAREA(130),A(IA,IA),AINV(IA,IA)
      DATA WKAREA/130*0./
      IDGT=10
      CALL LINV2F (A,N,IA,AINV,IDGT,WKAREA,IER)
      IF (IER.EQ.34) GO TO 25
      IF (IER.EQ.131) GO TO 30
      IF (IER.EQ.129) GO TO 35
      RETURN
   25 PRINT*, 'ACCURACY TEST FAILED...DATA MAY BE SUSPECT'
      RETURN
   30 PRINT*, 'MATRIX TOO ILL-CONDITIONED FOR ITERATIVE IMPROVEMENT TO B
     *E EFFECTIVE'
      RETURN 1
   35 PRINT*, 'MATRIX IS SINGULAR AND CANNOT BE INVERTED'
      RETURN 1
      END
C **********************************************************************
C **********************************************************************
C  OPTIONS #0 THRU #9
C **********************************************************************
C **********************************************************************
      OVERLAY (1,0)
      PROGRAM OPTO
      INTEGER P,PD,DATA
      CHARACTER LFN*30,DMATRX,ACT,SEN
      COMMON /B  1 / NOPT,IERR,IPLANT,IFLAG(40)
      COMMON /B  2 / N,M,P,ND,MD,PD
      COMMON /B  3 / GO(10,10)
      COMMON /B  4 / A(10,10),B(10,10),C(10,10)
      COMMON /B  5 / D(10,10)
      COMMON /B  6 / DMATRX,ACT,SEN
      COMMON /B  7A/ NA(10),AA(10,2,2),BA(10,2),CA(10,2)
      COMMON /B  7S/ NS(10),AS(10,2,2),BS(10,2),CS(10,2)
C ----------------------------------------------------------------------
      IERR=2
      GO TO (2001,2002,2003,2004,2005,2006,2007,2008,2009)NOPT
C **********************************************************************
C  OPTION #0
```

```
      PRINT*, 'PLANT INPUT OPTIONS:'
      PRINT*, '     0.  LIST OPTIONS 0 THRU 9'
      PRINT*, '     1.  ENTER G(0) MATRIX'
      PRINT*, '     2.  ENTER # OF STATES, INPUTS & OUTPUTS (N,M,P)'
      PRINT*, '     3.  ENTER PLANT A, B, C, & D MATRICES'
      PRINT*, '     4.  ENTER ACTUATOR STATE EQUATION MATRIX DATA'
      PRINT*, '     5.  ENTER SENSOR STATE EQUATION MATRIX DATA'
      PRINT*, '     6.  OPTION RESERVED'
      PRINT*, '     7.  OPTION RESERVED'
      PRINT*, '     8.  COPY G(0) INFO FROM LOCAL FILE'
      PRINT*, '     9.  COPY PLANT, ACTUATOR & SENSOR INFO FROM LOCAL FI
     *LE'
      GO TO 8001
C *********************************************************************
C  OPTION #1
 2001 PRINT*, 'THIS OPTION ENTERS THE STEADY STATE TRANSFER FUNCTION MAT
     *RIX, G(0)'
      PRINT*, '  '
      PRINT*, 'PLEASE ENTER THE NUMBER OF INPUTS AND OUTPUTS >'
      READ*, M,P
      IF (M.NE.P) THEN
          IERR=40
          GO TO 9001
      ENDIF
      PRINT*, '  '
      PRINT*, '  '
      PRINT*, 'ENTER G(0) MATRIX...',P,' ROWS WITH ',M,' ELEMENTS EACH'
      DO 260 I=1,P
      PRINT*, 'ROW ',I,' >'
  260 READ*, (GO(I,J),J=1,M)
      IPLANT=2
      INTER=1
      IFLAG(1)=1
      GO TO 8001
C *********************************************************************
C  OPTION #2
 2002 PRINT*, 'THIS OPTION SETS THE NUMBER OF STATES, INPUTS & OUTPUTS'
      PRINT '(/A)', ' ENTER NUMBER OF STATES, INPUTS AND OUTPUTS >'
      READ*, N,M,P
      IFLAG(2)=1
      GO TO 8001
C *********************************************************************
C  OPTION #3
 2003 PRINT*, 'THIS OPTION ENTERS THE PLANT A,B,C, AND D MATRICES'
      PRINT*, '  '
      IF (IFLAG(IERR).EQ.0) GO TO 9001
  265 PRINT*, 'ENTER "A" MATRIX...',N,' ROWS WITH ',N,' ELEMENTS EACH'
      DO 275 I=1,N
      PRINT*, 'ROW ',I,' >'
  275 READ*, (A(I,J),J=1,N)
```

```
      PRINT 4015
      CALL MATPR (A,N,N)
      CALL ANSWER (*265,*8001)
  280 PRINT*, 'ENTER "B" MATRIX...',N,' ROWS WITH ',M,' ELEMENTS EACH'
      DO 285 I=1,N
      PRINT*, 'ROW ',I,' >'
  285 READ*, (B(I,J),J=1,M)
      PRINT 4020
      CALL MATPR (B,N,M)
      CALL ANSWER (*280,*8001)
  290 PRINT*, 'ENTER "C" MATRIX...',P,' ROWS WITH ',N,' ELEMENTS EACH'
      DO 295 I=1,P
      PRINT*, 'ROW ',I,' >'
  295 READ*, (C(I,J),J=1,N)
      PRINT 4030
      CALL MATPR (C,P,N)
      CALL ANSWER (*290,*8001)
  296 PRINT*, 'IS THERE A "D" MATRIX...YES OR NO...>'
      READ '(A)', DMATRX
      IF (DMATRX.NE.'Y') THEN
          DO 297 I=1,P
          DO 297 J=1,M
  297     D(I,J)=0.
          GO TO 310
      ENDIF
      PRINT*, ' '
      PRINT*, 'ENTER "D" MATRIX...',P,' ROWS WITH ',M,' ELEMENTS EACH'
      DO 300 I=1,P
      PRINT*, 'ROW ',I,' >'
  300 READ*, (D(I,J),J=1,M)
      PRINT 4035
      CALL MATPR (D,P,M)
      CALL ANSWER (*296,*8001)
  310 CONTINUE
      IFLAG(3)=1
      IPLANT=1
      GO TO 8001
C *********************************************************************
C  OPTION #4
 2004 PRINT*, 'THIS OPTION ENTERS THE ACTUATOR STATE EQUATION DATA'
      PRINT*, ' '
      IF (IFLAG(IERR).EQ.0) GO TO 9001
      PRINT*, 'ENTER "0" TO ELIMINATE ACTUATORS'
      PRINT*, 'ENTER "1" TO SET ACTUATOR VALUES...>'
      READ*, ISKIP
      IF (ISKIP.EQ.0) GO TO 351
      PRINT*, 'ENTER NUMBER OF STATES OF EACH OF THE ',M,' ACTUATORS >'
      READ*, (NA(I),I=1,M)
      PRINT*, ' '
      DO 350 K=1,M
      PRINT*, 'ENTER THE DATA FOR ACTUATOR #',K,'...'
      PRINT*, '  '
      PRINT*, 'ENTER THE "A" MATRIX...',NA(K),' ROWS WITH ',NA(K),' ELEM
     *ENTS EACH'
      DO 340 I=1,NA(K)
      PRINT*, 'ROW ',I,' >'
  340 READ*, (AA(K,I,J),J=1,NA(K))
      PRINT*, '  '
      PRINT*, 'ENTER THE "B" COLUMN...',NA(K),' ELEMENTS'
      PRINT*, '"B" COLUMN >'
      READ*, (BA(K,I),I=1,NA(K))
      PRINT*, '  '
      PRINT*, 'ENTER THE "C" ROW...',NA(K),' ELEMENTS'
      PRINT*, '"C" ROW >'
      READ*, (CA(K,I),I=1,NA(K))
      PRINT '(/)'
  350 CONTINUE
      ACT='Y'
      IFLAG(4)=1
      GO TO 8001
C -------------------ELIMINATE ACTUATORS-----------------------------
  351 DO 352 K=1,10
      NA(K)=1
      DO 352 I=1,2
      BA(K,I)=9999.
      CA(K,I)=1.
      DO 352 J=1,2
  352 AA(K,I,J)=-9999.
      ACT='N'
      IFLAG(4)=0
      GO TO 8001
C *******************************************************************
C  OPTION #5
 2005 PRINT*, 'THIS OPTION ENTERS THE SENSOR STATE EQUATION DATA'
      PRINT*, '  '
      IF (IFLAG(IERR).EQ.0) GO TO 9001
      PRINT*, 'ENTER "0" TO ELIMINATE SENSORS'
      PRINT*, 'ENTER "1" TO SET SENSOR VALUES...>'
      READ*, ISKIP
      IF (ISKIP.EQ.0) GO TO 376
      PRINT*, 'ENTER NUMBER OF STATES OF EACH OF THE ',P,' SENSORS >'
      READ*, (NS(I),I=1,P)
      PRINT*, '  '
      DO 375 K=1,P
      PRINT*, 'ENTER THE DATA FOR SENSOR #',K,'...'
      PRINT*, '  '
      PRINT*, 'ENTER THE "A" MATRIX...',NS(K),' ROWS WITH ',NS(K),' ELEM
     *ENTS EACH'
      DO 360 I=1,NS(K)
      PRINT*, 'ROW ',I,' >'
  360 READ*, (AS(K,I,J),J=1,NS(K))
      PRINT*, '  '
      PRINT*, 'ENTER THE "B" COLUMN...',NS(K),' ELEMENTS'
      PRINT*, '"B" COLUMN >'
      READ*, (BS(K,I),I=1,NS(K))
      PRINT*, '  '
      PRINT*, 'ENTER THE "C" ROW...',NS(K),' ELEMENTS'
      PRINT*, '"C" ROW >'
      READ*, (CS(K,I),I=1,NS(K))
      PRINT '(/)'
  375 CONTINUE
      SEN='Y'
      IFLAG(5)=1
      GO TO 8001
C --------------------ELIMINATE SENSORS--------------------------------
  376 DO 377 K=1,10
      NS(K)=1
      DO 377 I=1,2
      BS(K,I)=9999.
      CS(K,I)=1.
      DO 377 J=1,2
  377 AS(K,I,J)=-9999.
      SEN='N'
      IFLAG(5)=0
      GO TO 8001
C *********************************************************************
C  OPTION #6
 2006 IERR=44
      GO TO 9001
C *********************************************************************
C  OPTION #7
 2007 IERR=44
      GO TO 9001
C *********************************************************************
C  OPTION #8
 2008 PRINT '(A/)', ' THIS OPTION COPIES G(0) FROM A LOCAL DATA FILE'
  380 PRINT*, 'ENTER LOCAL FILE NAME THAT HOLDS G(0) INFORMATION'
      PRINT*, '                      >'
      READ '(A)', LFN
      PRINT*, 'LOCAL FILE NAME IS >',LFN
      CALL ANSWER (*380,*8001)
      DATA=10
      OPEN (DATA,FILE=LFN)
      REWIND DATA
      READ (DATA,*) IPLANT
      IERR=8
      IF (IPLANT.EQ.1) GO TO 9001
      READ (DATA,*) M,P
      DO 385 I=1,P
  385 READ (DATA,*) (GO(I,J),J=1,M)
```

```
      PRINT '(/A)', ' DATA COPY COMPLETE FOR OPTION #1'
      IFLAG(1)=1
      INTER=2
      CLOSE (DATA,STATUS='KEEP')
      GO TO 8001
C ******************************************************************
C  OPTION #9
 2009 PRINT*, 'THIS OPTION COPIES STATE EQN INFO FROM A LOCAL DATA FILE'
      PRINT*, '  '
  390 PRINT*, 'ENTER LOCAL FILE NAME THAT HOLDS STATE EQN INFORMATION'
      PRINT*, '                              >'
      READ '(A)', LFN
      PRINT*, 'LOCAL FILE NAME IS >',LFN
      CALL ANSWER (*390,*8001)
      DATA=10
      OPEN (DATA,FILE=LFN)
      REWIND DATA
      READ (DATA,*) IPLANT
      IERR=9
      IF (IPLANT.EQ.0) PRINT '(/A/)', ' DATA MAY BE INCOMPLETE...CHECK P
     *RINTOUTS'
      IF (IPLANT.EQ.2) GO TO 9001
      READ (DATA,*) N,M,P
      DO 395 I=1,N
  395 READ (DATA,*) (A(I,J),J=1,N)
      DO 400 I=1,N
  400 READ (DATA,*) (B(I,J),J=1,M)
      DO 405 I=1,P
  405 READ (DATA,*) (C(I,J),J=1,N)
      READ (DATA,'(A)') DMATRX
      IF (DMATRX.EQ.'N') GO TO 430
      DO 410 I=1,P
  410 READ (DATA,*) (D(I,J),J=1,M)
  430 READ (DATA,'(A)') ACT
      IF (ACT.EQ.'N') GO TO 441
      READ (DATA,*) (NA(I),I=1,M)
      DO 440 K=1,M
      DO 435 I=1,NA(K)
  435 READ (DATA,*) (AA(K,I,J),J=1,NA(K))
      READ (DATA,*) (BA(K,I),I=1,NA(K))
      READ (DATA,*) (CA(K,I),I=1,NA(K))
  440 CONTINUE
  441 READ (DATA,'(A)') SEN
      IF (SEN.EQ.'N') GO TO 465
      READ (DATA,*) (NS(I),I=1,P)
      DO 450 K=1,P
      DO 445 I=1,NS(K)
  445 READ (DATA,*) (AS(K,I,J),J=1,NS(K))
      READ (DATA,*) (BS(K,I),I=1,NS(K))
      READ (DATA,*) (CS(K,I),I=1,NS(K))
  450 CONTINUE
  465 PRINT*, 'DATA COPY COMPLETE FOR OPTIONS'
      IF (ACT.EQ.'Y'.AND.SEN.EQ.'Y') PRINT*, '#2, #3, #4, #5'
      IF (ACT.EQ.'Y'.AND.SEN.EQ.'N') PRINT*, '#2, #3, #4'
      IF (ACT.EQ.'N'.AND.SEN.EQ.'Y') PRINT*, '#2, #3, #5'
      IF (ACT.EQ.'N'.AND.SEN.EQ.'N') PRINT*, '#2, #3'
      INTER=2
      DO 480 I=2,5
  480 IFLAG(I)=1
      CLOSE (DATA,STATUS='KEEP')
      GO TO 8001
C **********************************************************************
C **********************************************************************
C  FORMAT STATEMENTS
C **********************************************************************
 4015 FORMAT (/1X,'PLANT MATRIX A...'/)
 4020 FORMAT (/1X,'PLANT MATRIX B...'/)
 4030 FORMAT (/1X,'OUTPUT MATRIX C...'/)
 4035 FORMAT (/1X,'DIRECT OUTPUT MATRIX D...'/)
 4050 FORMAT (1X,1F10.4,23X,1F10.4,11X,1F10.4)
C ----------------------------------------------------------------------
 8001 IERR=0
 9001 CONTINUE
      END
C **********************************************************************
C **********************************************************************
C  OPTIONS #10 THRU #19 -- LESS DESIGN OPTIONS
C **********************************************************************
C **********************************************************************
      OVERLAY (2,0)
      PROGRAM OPT10
      CHARACTER METHOD,LFN*30
      REAL K0,K1,MM
      INTEGER P,PD,DATB
      COMMON /B  1 / NOPT,IERR,IPLANT,IFLAG(40)
      COMMON /B  2 / N,M,P,ND,MD,PD
      COMMON /B  8 / ALPHA,EPSLON,GAMA(10,10)
      COMMON /B  8A/ METHOD
      COMMON /B  9 / K0(10,10),K1(10,10)
      COMMON /B 10 / NG,GN(10)
      COMMON /B 11 / MM(10,10)
      DIMENSION VV(10,10),WW(10,10),ZZ(10,10)
      DATA VV,WW,ZZ/300*0./
C ----------------------------------------------------------------------
      IERR=2
      NOPT=NOPT-10
      GO TO (2011,2012,2013,8002,2015,2016,2017,8002,2019) NOPT
C **********************************************************************
C  OPTION #10
      PRINT*, 'DESIGN PARAMETER INPUT OPTIONS:'
      PRINT*, '    10.  LIST OPTIONS 10 THRU 19'
      PRINT*, '    11.  ENTER ALPHA'
      PRINT*, '    12.  ENTER SIGMA WEIGHTING MATRIX'
      PRINT*, '    13.  ENTER EPSILON (SIGMA MATRIX MULTIPLIER)'
      PRINT*, '    14.  RUN DESIGN...UNKNOWN, REGULAR & IRREGULAR PLANTS
     *'
      PRINT*, '    15.  OPTION RESERVED'
      PRINT*, '    16.  OPTION RESERVED'
      PRINT*, '    17.  OPTION RESERVED'
      PRINT*, '    18.  MEASUREMENT MATRIX FORMATION...OR'
      PRINT*, '         ROW & COLUMN OPERATIONS ON C*B OR OTHER MATRIX'
      PRINT*, '    19.  COPY DESIGN PARAMETERS FROM LOCAL FILE'
      GO TO 8002
C **********************************************************************
C  OPTION #11
 2011 PRINT*, 'THIS OPTION SETS THE PROPORTION OF INTEGRAL AND DIRECT FE
     *EDBACK'
      PRINT*, '  '
      PRINT*, 'ENTER ALPHA >'
      READ*, ALPHA
      IFLAG(11)=1
      GO TO 8002
C **********************************************************************
C  OPTION #12
 2012 PRINT*, 'THIS OPTION SETS THE WEIGHTING BETWEEN OUTPUT CHANNELS'
      PRINT*, '  '
      IF (IPLANT.EQ.2) GO TO 501
      IF (IFLAG(IERR).EQ.0) GO TO 9002
  501 PRINT*, 'ENTER SIGMA WEIGHTING MATRIX...',P,' DIAGONAL ELEMENTS ON
     *LY'
      PRINT*, 'DIAGONAL ELEMENTS >'
      READ*, (GAMA(I,I),I=1,P)
      IFLAG(12)=1
      GO TO 8002
C **********************************************************************
C  OPTION #13
 2013 PRINT*, 'THIS OPTION SETS THE WEIGHTING MATRIX MULTIPLIER'
      PRINT*, '  '
      PRINT*, 'ENTER SIGMA MATRIX SCALAR MULTIPLIER, EPSILON >'
      READ*, EPSLON
      IFLAG(13)=1
      GO TO 8002
C **********************************************************************
C  OPTION #14 IS AN OVERLAY
C **********************************************************************
C  OPTION #15
 2015 IERR=44
      NOPT=NOPT+10
      GO TO 9002
C **********************************************************************
```

```
C  OPTION #16
 2016 IERR=44
      NOPT=NOPT+10
      GO TO 9002
C **********************************************************************
C  OPTION #17
 2017 IERR=44
      NOPT=NOPT+10
      GO TO 9002
C **********************************************************************
C  OPTION #18 IS AN OVERLAY
C **********************************************************************
C  OPTION #19
 2019 PRINT*, 'THIS OPTION READS DESIGN PARAMETERS FROM A LOCAL DATA FIL
     *E'
      IF (IFLAG(IERR).EQ.0) GO TO 9002
  505 PRINT '(/A)', ' ENTER LOCAL FILE NAME THAT HOLDS DESIGN PARAMETERS
     *'
      PRINT*, '                      >'
      READ '(A)', LFN
      PRINT*, 'LOCAL FILE NAME IS >',LFN
      CALL ANSWER (*505,*8002)
      DATB=35
      OPEN (DATB,FILE=LFN)
      REWIND DATB
      READ (DATB,'(A)') METHOD
      READ (DATB,*) ALPHA
      READ (DATB,*) (GAMA(I,I),I=1,P)
      READ (DATB,*) EPSLON
      DO 510 I=1,M
  510 READ (DATB,*) (K0(I,J),J=1,P)
      IF (ALPHA.EQ.1) THEN
          DO 515 I=1,M
          DO 515 J=1,P
  515     K1(I,J)=K0(I,J)
          GO TO 530
      ENDIF
      DO 520 I=1,M
  520 READ (DATB,*) (K1(I,J),J=1,P)
  530 IF (METHOD.EQ.'I') THEN
          DO 540 I=1,P
  540     READ (DATB,*) (MM(I,J),J=1,N-M)
      ENDIF
      PRINT*, 'DATA COPY COMPLETE FOR OPTIONS'
      IF (METHOD.NE.'I') PRINT*, '#11, #12, #13, #14'
      IF (METHOD.EQ.'I') PRINT*, '#11, #12, #13, #14, #18'
      DO 565 IERR=11,14
  565 IFLAG(IERR)=1
      IF (METHOD.EQ.'I') IFLAG(18)=1
      CLOSE (DATB,STATUS='KEEP')
```

```
C ---------------------------------------------------------------------
 8002 IERR=0
 9002 CONTINUE
      END
C *********************************************************************
C *********************************************************************
C  OPTION #14 -- UNKNOWN PLANTS
C *********************************************************************
C *********************************************************************
      OVERLAY (3,0)
      PROGRAM OPT14U
      INTEGER P,PD
      REAL K0,K1
      CHARACTER METHOD,LFN*30
      COMMON /B  1 / NOPT,IERR,IPLANT,IFLAG(40)
      COMMON /B  2 / N,M,P,ND,MD,PD
      COMMON /B  3 / G0(10,10)
      COMMON /B  4 / A(10,10),B(10,10),C(10,10)
      COMMON /B  5 / D(10,10)
      COMMON /B  8 / ALPHA,EPSLON,GAMA(10,10)
      COMMON /B  8A/ METHOD
      COMMON /B  9 / K0(10,10),K1(10,10)
      DIMENSION CM(10,10),VV(10,10),WW(10,10),ZZ(10,10)
      DATA CM,VV,WW,ZZ/400*0./
      DO 601 I=1,P
      DO 601 J=1,N
  601 CM(I,J)=C(I,J)
      PRINT*, 'THIS OPTION COMPUTES K0 & K1 FOR UNKNOWN PLANTS'
      DO 603 IERR=11,13
  603 IF (IFLAG(IERR).EQ.0) GO TO 9003
      IF (IPLANT.EQ.2) GO TO 610
      DO 605 IERR=2,3
  605 IF (IFLAG(IERR).EQ.0) GO TO 9003
      IPLANT=1
  610 CONTINUE
      IF (IPLANT.EQ.2) GO TO 650
C --------------------CREATE AI * B-----------------------------------
      CALL INVERT (A,VV,N,ND,*695)
      DO 620 I=1,N
      DO 620 J=1,M
      S=0.
      DO 615 K=1,N
  615 S=S+VV(I,K)*B(K,J)
      ZZ(I,J)=S
  620 CONTINUE
C --------------------CREATE G(0)--------------------------------------
      DO 630 I=1,P
      DO 630 J=1,M
      S=0.
      DO 625 K=1,N
  625 S=S+C(I,K)*ZZ(K,J)
      GO(I,J)=D(I,J)-S
  630 CONTINUE
      LFN='G(0) MATRIX PRINTOUT'
      CALL QPRINT (LFN,*635)
      PRINT '(/A/)', ' G(0) MATRIX...'
      CALL MATPR (GO,P,M)
C --------------------CREATE GM(0)------------------------------------
  635 DO 645 I=1,P
     DO 645 J=1,M
      S=0.
      DO 640 K=1,N
  640 S=S+CM(I,K)*ZZ(K,J)
      WW(I,J)=D(I,J)-S
  645 CONTINUE
C --------------------ENTRY FOR G(0) FROM FILE------------------------
  650 IF (IPLANT.EQ.2) THEN
          DO 655 I=1,P
          DO 655 J=1,M
  655     WW(I,J)=GO(I,J)
      ENDIF
C --------------------CREATE K0 & K1----------------------------------
  675 CALL INVERT (GO,WW,M,MD,*705)
      DO 685 I=1,M
      DO 685 J=1,P
      S=0.
      DO 680 K=1,P
  680 S=S+WW(I,K)*GAMA(K,J)
      K1(I,J)=S*EPSLON
      K0(I,J)=ALPHA*S*EPSLON
  685 CONTINUE
      PRINT '(/A)', ' K0 & K1 MATRICES FORMED'
C -------------------------------------------------------------------
      IFLAG(14)=1
      METHOD='U'
      GO TO 8003
  695 IERR=36
      GO TO 9003
  705 IERR=31
      GO TO 9003
 8003 IERR=0
 9003 IF (IERR.NE.0) METHOD='X'
      END
C ******************************************************************
C ******************************************************************
C  OPTION #14 -- REGULAR PLANTS
C ******************************************************************
C ******************************************************************
      OVERLAY (4,0)
      PROGRAM OPT14R
      REAL K0,K1
      INTEGER P,PD
      CHARACTER METHOD,LFN*30
      COMMON /B  1 / NOPT,IERR,IPLANT,IFLAG(40)
      COMMON /B  2 / N,M,P,ND,MD,PD
      COMMON /B  4 / A(10,10),B(10,10),C(10,10)
      COMMON /B  8 / ALPHA,EPSLON,GAMA(10,10)
      COMMON /B  8A/ METHOD
      COMMON /B  9 / K0(10,10),K1(10,10)
      DIMENSION VV(10,10),WW(10,10),ZZ(10,10)
      DATA VV,WW,ZZ/300*0./
      PRINT*, 'THIS OPTION COMPUTES K0 & K1 FOR REGULAR PLANTS'
      DO 800 IERR=11,13
  800 IF (IFLAG(IERR).EQ.0) GO TO 9004
      DO 805 IERR=2,3
  805 IF (IFLAG(IERR).EQ.0) GO TO 9004
C --------------------CREATE C*B, BI*CI & BI*CI*GAMA-------------------
      CALL VMULFF (C,B,P,N,M,PD,ND,VV,PD,IER)
      IF (IER.EQ.129) THEN
          IERR=39
          GO TO 9004
      ENDIF
      LFN='C*B MATRIX PRINTOUT'
      CALL QPRINT (LFN,*810)
      PRINT '(/A/)', ' C*B MATRIX...'
      CALL MATPR (VV,P,M)
  810 CALL INVERT (VV,ZZ,P,PD,*835)
      CALL VMULFF (ZZ,GAMA,M,P,P,MD,PD,WW,MD,IER)
      IF (IER.EQ.129) THEN
        IERR=39
        GO TO 9004
      ENDIF
C --------------------CREATE K0 & K1-----------------------------------
      DO 815 I=1,M
      DO 815 J=1,P
      K0(I,J)=WW(I,J)*ALPHA*EPSLON
      K1(I,J)=WW(I,J)*EPSLON
  815 CONTINUE
      PRINT '(/A)', ' K0 & K1 MATRICES FORMED'
C --------------------------------------------------------------------
      IFLAG(14)=1
      IPLANT=1
      METHOD='R'
      GO TO 8004
  835 IERR=32
      GO TO 9004
 8004 IERR=0
 9004 IF (IERR.NE.0) METHOD='X'
      END
C ********************************************************************
```

```
C **********************************************************************
C  OPTION #14 -- IRREGULAR PLANTS
C **********************************************************************
C **********************************************************************
      OVERLAY (5,0)
      PROGRAM OPT14I
      REAL K0,K1,MM
      INTEGER P,PD
      CHARACTER METHOD,LFN*30
      COMMON /B  1 / NOPT,IERR,IPLANT,IFLAG(40)
      COMMON /B  2 / N,M,P,ND,MD,PD
      COMMON /B  4 / A(10,10),B(10,10),C(10,10)
      COMMON /B  8 / ALPHA,EPSLON,GAMA(10,10)
      COMMON /B  8A/ METHOD
      COMMON /B  9 / K0(10,10),K1(10,10)
      COMMON /B 11 / MM(10,10)
      DIMENSION VV(10,10),WW(10,10),ZZ(10,10),VWZ(10,10)
      DATA VV,WW,ZZ,VWZ/400*0./
C ----------------------------------------------------------------------
      PRINT '(A/)', ' THIS OPTION COMPUTES K0 & K1 FOR IRREGULAR PLANTS'
      DO 900 IERR=11,13
  900 IF (IFLAG(IERR).EQ.0) GO TO 9005
      DO 905 IERR=2,3
  905 IF (IFLAG(IERR).EQ.0) GO TO 9005
      IPLANT=1
      IF (N.EQ.M) THEN
          IERR=42
          GO TO 9005
      ENDIF
C --------------------INITIALIZE MEASUREMENT MATRIX--------------------
      IF (IFLAG(18).EQ.1) THEN
          PRINT *, 'ENTER "1" TO REINITIALIZE MEASUREMENT MATRIX, ELSE E
     *NTER "0" >'
          GO TO 906
      ENDIF
      PRINT*, 'ENTER "0" IF MEASUREMENT MATRIX EXISTS IN MEMORY'
      PRINT*, 'ENTER "1" TO INITIALIZE MATRIX IN THIS OPTION...>'
  906 READ*, ISKIP
      IF (ISKIP.EQ.0) THEN
          IERR=18
          IF (IFLAG(IERR).EQ.0) GO TO 9005
      ENDIF
      IF (ISKIP.EQ.1) THEN
          PRINT*, '  '
  910     PRINT*, 'ENTER M MATRIX...',P,' ROWS WITH ',N-M,' COLUMNS'
          DO 915 I=1,P
          PRINT*, 'ROW ',I,' >'
  915     READ*, (MM(I,J),J=1,N-M)
          PRINT '(/A/)', ' MEASUREMENT MATRIX...'
          CALL MATPR (MM,P,N-M)
          CALL ANSWER (*910,*8005)
          IFLAG(18)=1
      ENDIF
C --------------------CREATE [F1,F2] FROM [C1+MA11,C2+MA12]---- -- -------
      DO 925 I=1,P
      DO 925 J=1,N
      S=0.
      DO 920 K=1,N-M
  920 S=S+MM(I,K)*A(K,J)
      WW(I,J)=C(I,J)+S
  925 CONTINUE
      LFN='F=[F1,F2] MATRIX'
      CALL QPRINT (LFN,*930)
      PRINT '(/A/)', ' F=[F1,F2] MATRIX...'
      CALL MATPR (WW,P,N)
C --------------------CREATE F2, F2I & F2I*GAMA-------------------------
  930 DO 935 I=1,P
      DO 935 J=1,M
  935 ZZ(I,J)=WW(I,N-M+J)
      CALL INVERT (ZZ,WW,P,PD,*990)
      CALL VMULFF (WW,GAMA,M,P,P,MD,PD,ZZ,MD,IER)
      IF (IER.EQ.129) THEN
        IERR=39
        GO TO 9005
      ENDIF
C --------------------CREATE B2, B2I, B2I*F2I*GAMA----------------------
      DO 950 I=1,M
      DO 950 J=1,M
  950 WW(I,J)=B(N-M+I,J)
      CALL INVERT (WW,VWZ,M,MD,*985)
      CALL VMULFF (VWZ,ZZ,M,M,P,MD,MD,WW,MD,IER)
      IF (IER.EQ.129) THEN
        IERR=39
        GO TO 9005
      ENDIF
C --------------------CREATE K0 & K1------------------------------------
      DO 965 I=1,P
      DO 965 J=1,P
      K0(I,J)=WW(I,J)*ALPHA*EPSLON
  965 K1(I,J)=WW(I,J)*EPSLON
      PRINT '(/A)', ' K0 & K1 MATRICES FORMED'
C ----------------------------------------------------------------------
      IFLAG(14)=1
      METHOD='I'
      GO TO 8005
  985 IERR=33
      GO TO 9005
  990 IERR=34
      GO TO 9005
 8005 IERR=0
 9005 IF (IERR.NE.0) METHOD='X'
      END
C *********************************************************************
C *********************************************************************
C  OPTION #18
C *********************************************************************
C *********************************************************************
      OVERLAY (6,0)
      PROGRAM OPT18
      INTEGER P,PD
      REAL MM
      COMMON /B  1 / NOPT,IERR,IPLANT,IFLAG(40)
      COMMON /B  2 / N,M,P,ND,MD,PD
      COMMON /B  4 / A(10,10),B(10,10),C(10,10)
      COMMON /B 11 / MM(10,10)
      DIMENSION VV(10,10),WW(10,10),ZZ(10,10),VWZ(10,10)
      DATA VV,WW,ZZ,VWZ/400*0./
C ---------------------------------------------------------------------
      PRINT*, 'THIS OPTION CREATES A MEASUREMENT MATRIX...OR'
      PRINT '(A/)', ' PERFORMS ROW AND COLUMN OPTIONS ON A MATRIX'
      PRINT*, 'ENTER 1 TO CALCULATE MEASUREMENT MATRIX FROM C.E.S.A. "K"
     *MATRIX'
      PRINT*, 'TYPE 2 TO OPERATE ON C * B '
      PRINT*, 'TYPE 3 TO OPERATE ON ANY OTHER MATRIX'
      READ*, ISKIP
      PRINT*, ' '
C --------------------CREATE M MATRIX FROM C.E.S.A. INPUT---------------
      IF (ISKIP.EQ.1) THEN
          IF (IFLAG(5).EQ.0) THEN
              IERR=5
              GO TO 9006
          ENDIF
 1000     PRINT*, 'ENTER K1 MATRIX FROM CESA OPTION #38...',M,' ROWS WIT
     *H ',N-M,' COLUMNS'
          DO 1005 I=1,M
          PRINT*, 'ROW ',I,' >'
 1005     READ*, (VWZ(I,J),J=1,N-M)
          CALL MATPR (VWZ,M,N-M)
          CALL ANSWER (*1000,*8006)
C --------------------CREATE C1 & C2------------------------------------
          DO 1015 I=1,P
          DO 1010 J=1,N-M
 1010     MM(I,J)=C(I,J)
          DO 1015 J=1,M
 1015     WW(I,J)=C(I,N-M+J)
C ---------------------CREATE AMAT = [ C1 - C2 * CESA ]-----------------
          DO 1025 I=1,P
          DO 1025 J=1,N-M
          S=0.
          DO 1020 K=1,M
```

```
1020      S=WW(I,K)*VWZ(K,J)
1025      ZZ(I,J)=MM(I,J)-S
C --------------------CREATE A11 & A12-------------------------------
          DO 1035 I=1,N-M
          DO 1030 J=1,N-M
1030      MM(I,J)=A(I,J)
          DO 1035 J=1,M
1035      WW(I,J)=A(I,N-M+J)
C --------------------CREATE BMAT = [ A12 * CESA - A11 ]---------------
          DO 1045 I=1,N-M
          DO 1045 J=1,N-M
          S=0.
          DO 1040 K=1,M
1040      S=WW(I,K)*VWZ(K,J)
1045      VV(I,J)=S-MM(I,J)
C --------------------CREATE MEASUREMENT MATRIX-------------------------
          CALL INVERT (VV,WW,N-M,ND,*1097)
          CALL VMULFF (ZZ,WW,P,N-M,N-M,PD,ND,MM,PD,IER)
          IF (IER.EQ.129) THEN
              IERR=39
              GO TO 9006
          ENDIF
          IFLAG(18)=1
          PRINT '(/A)', ' MEASUREMENT MATRIX FORMED'
          GO TO 8006
C --------------------CREATE C * B--------------------------------------
        ELSEIF (ISKIP.EQ.2) THEN
          IF (IFLAG(5).EQ.0) THEN
              IERR=5
              GO TO 9006
          ENDIF
          DO 1055 I=1,P
          DO 1055 J=1,M
          S=0.
          DO 1050 K=1,N
1050      S=S+C(I,K)*B(K,J)
          WW(I,J)=S
1055      CONTINUE
          PRINT '(/A/)', ' C * B MATRIX...'
          CALL MATPR (WW,P,M)
          II=P
          JJ=M
          GO TO 1070
C --------------------ENTER OTHER MATRIX--------------------------------
        ELSEIF (ISKIP.EQ.3) THEN
          PRINT*, 'ENTER MATRIX''S # OF ROWS AND # OF COLUMNS >'
          READ*, II,JJ
1060      PRINT '(/A)', ' ENTER MATRIX...',II,' ROWS WITH ',JJ,' COLUMNS
     *'
          DO 1065 I=1,II
          PRINT*, 'ROW ',II,' >'
 1065     READ*, (WW(I,J),J=1,JJ)
          PRINT*, '(/A/)', ' MATRIX IS...'
          CALL MATPR (WW,II,JJ)
          CALL ANSWER (*1060,*8006)
      ENDIF
C -------------------ROW & COLUMN OPERATIONS-------------------------
 1070 PRINT*, 'TO DO COLUMN AND ROW OPERATIONS ON MATRIX...'
 1075 PRINT*, '    ENTER 1 FOR ROW OPERATIONS'
      PRINT*, '    ENTER 2 FOR COLUMN OPERATIONS'
      PRINT*, '    ENTER 3 FOR ROW MULTIPLICATION'
      PRINT*, '    ENTER 4 FOR COLUMN MULTIPLICATION'
      PRINT*, '    ENTER 0 TO EXIT > '
      READ*, ISKIP
      PRINT*, ' '
      IF (ISKIP.EQ.1) THEN
          PRINT*, 'ENTER X,Y,Z SUCH THAT...'
          PRINT*, '    (X,ROW) * (Y,FACTOR) + (Z,ROW) = (NEW Z,ROW)'
          READ*, IROW1,FACT,IROW2
          DO 1080 J=1,JJ
 1080     WW(IROW2,J)=WW(IROW2,J)+FACT*WW(IROW1,J)
      ELSEIF (ISKIP.EQ.2) THEN
          PRINT*, 'ENTER X,Y,Z SUCH THAT...'
          PRINT*, '    (X,COLUMN) * (Y,FACTOR) + (Z,COLUMN) = (NEW Z,COL
     *UMN)'
          READ*, IROW1,FACT,IROW2
          DO 1085 I=1,II
 1085     WW(I,IROW2)=WW(I,IROW2)+FACT*WW(I,IROW1)
      ELSEIF (ISKIP.EQ.3) THEN
          PRINT*, 'ENTER X,Y SUCH THAT....'
          PRINT*, '    (X,ROW) * (Y,FACTOR) = (NEW X,ROW)'
          READ*, IROW,FACT
          DO 1090 J=1,JJ
 1090     WW(IROW,J)=FACT*WW(IROW,J)
      ELSEIF (ISKIP.EQ.4) THEN
          PRINT*, 'ENTER X,Y SUCH THAT....'
          PRINT*, '    (X,COLUMN) * (Y,FACTOR) = (NEW X,COLUMN)'
          READ*, IROW,FACT
          DO 1095 I=1,II
 1095     WW(I,IROW)=FACT*WW(I,IROW)
      ELSE
          GO TO 8006
      ENDIF
      PRINT '(/A/)', ' MATRIX NOW IS...'
      GO TO 1075
C -----------------------------------------------------------------
 1097 IERR=35
 8006 IERR=0
 9006 CONTINUE
      END
```

```
C **************************************************************************
C **************************************************************************
C  OPTIONS #20 THRU #29
C **************************************************************************
C **************************************************************************
      OVERLAY (7,0)
      PROGRAM OPT20
      CHARACTER STRING*10,METHOD,ACT,SEN,DMATRX,LFN*30
      REAL K0,K1,MM,MMAX
      INTEGER P,PD,DATC
      EXTERNAL CLPASS
      COMMON /B  1 / NOPT,IERR,IPLANT,IFLAG(40)
      COMMON /B  2 / N,M,P,ND,MD,PD
      COMMON /B  4 / A(10,10),B(10,10),C(10,10)
      COMMON /B  5 / D(10,10)
      COMMON /B  6 / DMATRX,ACT,SEN
      COMMON /B  7A/ NA(10),AA(10,2,2),BA(10,2),CA(10,2)
      COMMON /B  7S/ NS(10),AS(10,2,2),BS(10,2),CS(10,2)
      COMMON /B  8A/ METHOD
      COMMON /B  9 / K0(10,10),K1(10,10)
      COMMON /B 10 / NG,GN(10)
      COMMON /B 11 / MM(10,10)
      COMMON /B 12 / W(10),Q(10),U(10),YM(10)
      COMMON /B 12A/ X0(10),Z0(10),V(10)
      COMMON /B 12B/ ST,TT
      COMMON /B 13 / NT,YP(100,11,2)
      COMMON /B 13A/ UP(100,11,2)
      COMMON /B 13B/ ILIM,IMIN(10),IMAX(10),UMIN(10),UMAX(10)
      DIMENSION CM(10,10),F(10,10)
      DIMENSION IWORK(5),WORK(1150)
      DIMENSION X(50),Z(10)
      DIMENSION Y(10),E(10),AX(10),MMAX(10)
      DATA STRING/'1234567890'/
      DATA CM,F/200*0./
      DATA IWORK,AX,MMAX,WORK/5*0,1170*0./
      DATA X,Y,Z,E/80*0./
C --------------------------------------------------------------------------
      IERR=2
      DO 1101 I=1,P
      DO 1101 J=1,N
 1101 CM(I,J)=C(I,J)
      DO 1102 I=1,P
      DO 1102 J=1,P
      F(I,J)=0.0
 1102 IF (I.EQ.J) F(I,J)=1.0
      NOPT=NOPT-20
      GO TO (2021,2022,2023,2024,2025,2026,2027,2028,2029) NOPT
C **************************************************************************
C  OPTION #20
      PRINT*, 'SIMULATION OPTIONS:'
      PRINT*, '    20.  LIST OPTIONS 20 THRU 29'
      PRINT*, '    21.  SET STATE & INTEGRATOR INITIAL VALUES, X(0) & Z(
     *0)'
      PRINT*, '    22.  SET INPUT COMMAND VECTOR, V'
      PRINT*, '    23.  ENTER SAMPLE TIMES'
      PRINT*, '    24.  ENTER SIMULATION TIME'
      PRINT*, '    25.  ENTER CALCULATION STEP SIZE'
      PRINT*, '    26.  RUN SIMULATION'
      PRINT*, '    27.  SET CONTROL INPUT LIMITS'
      PRINT*, '    28.  OPTION RESERVED'
      PRINT*, '    29.  COPY SIMULATION PARAMETERS FROM LOCAL FILE'
      GO TO 8007
C ***********************************************************************
C  OPTION #21
 2021 PRINT*, 'THIS OPTION SETS THE INITIAL CONDITION VECTORS FOR THE ST
     *ATES & INTEGRATORS'
      PRINT*, '  '
      IF (IFLAG(IERR).EQ.0) GO TO 9007
      PRINT*, 'ENTER THE X(0) VECTOR OF ',N,' ELEMENTS'
      PRINT*, '>'
      READ*, (X0(I),I=1,N)
      PRINT*, '  '
      PRINT*, 'ENTER THE Z(0) VECTOR OF ',P,' ELEMENTS'
      PRINT*, '>'
      READ*, (Z0(I),I=1,P)
      IFLAG(21)=1
      GO TO 8007
C ***********************************************************************
C  OPTION #22
 2022 PRINT '(A/)', ' THIS OPTION SETS THE INPUT COMMAND VECTOR, V'
      IF (IFLAG(IERR).EQ.0) GO TO 9007
      PRINT*, 'ENTER THE V VECTOR OF ',P,' ELEMENTS'
      PRINT*, '"V" COLUMN >'
      READ*, (V(I),I=1,P)
      IFLAG(22)=1
      GO TO 8007
C ***********************************************************************
C  OPTION #23
 2023 PRINT*, 'THIS OPTION SETS THE SAMPLING TIME FOR EACH RUN'
      PRINT*, '  '
      PRINT*, 'ENTER NUMBER (MAX OF 2) OF SAMPLING TIMES >'
      READ*, NG
      PRINT*, 'ENTER ',NG,' SAMPLING TIME(S) >'
      READ*, (GN(I),I=1,NG)
      IFLAG(23)=1
      GO TO 8007
C ***********************************************************************
C  OPTION #24
 2024 PRINT*, 'THIS OPTION SETS THE TOTAL SIMULATION TIME'
      PRINT '(/A)', ' ENTER TOTAL TIME >'
      READ*, TT
      IFLAG(24)=1
      GO TO 8007
C ***********************************************************************
C  OPTION #25
 2025 PRINT*, 'THIS OPTION SETS THE CALCULATION STEP SIZE'
      PRINT '(/A)', ' ENTER STEP SIZE >'
      READ*, ST
      IFLAG(25)=1
      GO TO 8007
C ***********************************************************************
C  OPTION #26
 2026 PRINT*, 'THIS OPTION RUNS THE SIMULATION IF ALL DATA IS AVAILABLE'
      PRINT*, '  '
      IF (ACT.EQ.'N') PRINT*, 'SIMULATION INCLUDES NO ACTUATORS'
      IF (SEN.EQ.'N') PRINT*, 'SIMULATION INCLUDES NO SENSORS'
      IERR=14
      IF (IFLAG(IERR).EQ.1) GO TO 1105
      GO TO 9007
 1105 IF (IPLANT.NE.1) THEN
          DO 1107 IERR=2,3
 1107     IF (IFLAG(IERR).EQ.0) GO TO 9007
      ENDIF
      DO 1110 IERR=21,25
 1110 IF (IFLAG(IERR).EQ.0) GO TO 9007
      IF (ILIM.EQ.0) PRINT*, 'SIMULATION INCLUDES NO CONTROL LIMITS'
C -------------------TOTAL # OF STATES---------------------------------
      NO=0
      DO 1125 I=1,P
 1125 NO=NO+NS(I)
      DO 1130 I=1,M
 1130 NO=NO+NA(I)
      NN=N+NO
C --------------------LOOP FOR EACH SAMPLE TIME------------------------
      DO 1285 IG=1,NG
C --------------------INITIALIZATIONS----------------------------------
      S=0.
      T=0.
      DO 1135 I=1,M
      Z(I)=Z0(I)
      W(I)=0.
 1135 U(I)=0.
      DO 1140 I=1,P
 1140 Q(I)=0.
      DO 1145 I=1,N
 1145 X(I)=X0(I)
      DO 1150 I=1,NO
 1150 X(I+N)=0.
      PRINT*, '  '
      SAMT=GN(IG)
      PRINT*, ' '
      PRINT*, 'RUN TIME=',TT,'  STEP TIME=',ST,'  SAMPLE TIME=',SAMT
      PRINT*, ' '
      NIN=INT(TT/SAMT+.5)
      NTT=INT(SAMT/ST+.5)
      IF (SAMT.LT.ST) NTT=1
      NT=NIN*NTT
C --------------------SELECT SEQUENTIAL LIST PRINTING------------------
      PRINT*, 'THERE ARE ',NT,' TIME INCREMENTS'
      PRINT*, '   ENTER "0" TO SKIP SEQUENTIAL LISTING'
      PRINT*, '   ENTER "1" TO OBTAIN THIS DATA...>'
      READ*, ISKIP
      IF (ISKIP.EQ.0) PRINT '(/A)', ' ----CALCULATIONS IN PROGRESS----'
      PRINT*, ' '
C --------------------CHECK FOR NEED TO PACK DATA INTO 100 ELEMENTS-----
      IPACK=1
      IF (NT.GT.100) THEN
          PACK=MOD(NT,100)
          IF (PACK.LE.0.25) THEN
              IPACK=INT(NT/100)
          ELSE
              IPACK=(INT(NT/100)+1)
          ENDIF
      ENDIF
C --------------------LOOP FOR EACH SAMPLING PERIOD--------------------
      DO 1250 KT=1,NIN
C --------------------FORM Y MEASUREMENT VECTOR-----------------------
      DO 1160 I=1,P
      S=0.
      DO 1155 J=1,N
 1155 S=S+CM(I,J)*X(J)
      YM(I)=S
 1160 CONTINUE
C --------------------O.D.E. ROUTINE PRECISION------------------------
      RELERR=1.0E-4
      ABSERR=1.0E-4
C --------------------FORM ERROR VECTOR-------------------------------
      IF (METHOD.EQ.'I') THEN
          DO 1170 I=1,N-M
          S=0.
          DO 1165 J=1,N
 1165     S=S+A(I,J)*X(J)
 1170     AX(I)=S
          DO 1185 I=1,P
          S=0.
          DO 1175 K=1,N-M
 1175     S=S+MM(I,K)*AX(K)
          MMAX(I)=S
          S=0.
          DO 1180 J=1,P
 1180      S=S+F(I,J)*Q(J)
 1185      E(I)=V(I)-S-MMAX(I)
       ELSE
           DO 1195 I=1,P
           S=0.
           DO 1190 J=1,P
 1190      S=S+F(I,J)*Q(J)
 1195      E(I)=V(I)-S
       ENDIF
C --------------------FORM CONTROL LAW---------------------------------
       DO 1205 I=1,M
       S=0.
       DO 1200 J=1,P
       S=S+K0(I,J)*E(J)
 1200  S=S+K1(I,J)*Z(J)
       IF (METHOD.NE.'U') THEN
           U(I)=S/SAMT
       ELSE
           U(I)=S*SAMT
       ENDIF
 1205  CONTINUE
C --------------------CHECK AGAINST CONTROL LIMIT VALUE-----------------
       IF (ISKIP.NE.1) GO TO 1210
       PRINT*, '           T=',T
       IF (ILIM.EQ.0) GO TO 1210
       PRINT 4065, '*',(U(I),I=1,M)
 1210  DO 1215 I=1,M
       IF (U(I).LT.UMIN(I)) THEN
           U(I)=UMIN(I)
           IMIN(I)=IMIN(I)+1
       ELSEIF (U(I).GT.UMAX(I)) THEN
           U(I)=UMAX(I)
           IMAX(I)=IMAX(I)+1
       ENDIF
 1215  CONTINUE
C --------------------FORM INITIAL OUTPUT VALUE------------------------
       CALL YOUT (X,Y,W,C,D)
C --------------------INNER LOOP BASED ON STEP TIME--------------------
       DO 1240 JJ=1,NTT
C --------------------SET INDEX FOR INPUT/OUTPUT DATA MATRIX-----------
       IJ=JJ+(KT-1)*NTT
       PACK=REAL(IPACK)
       RIJ=REAL(IJ)
       IF (RIJ/PACK.NE.REAL(INT(IJ/IPACK))) THEN
           GO TO 1230
       ELSE
           IJ=IJ/IPACK
       ENDIF
       YP(IJ,1,IG)=T
       UP(IJ,1,IG)=T
```

```
      DO 1220 I=1,P
      YP(IJ,1+I,IG)=Y(I)
 1220 CONTINUE
      DO 1225 I=1,M
      UP(IJ,1+I,IG)=U(I)
 1225 CONTINUE
 1230 TOUT=T+ST
      IF (SAMT.LT.ST) TOUT=T+SAMT
      IF (ISKIP.NE.1) GO TO 1235
      PRINT 4065, 'U',(U(K),K=1,M)
      PRINT 4065, 'Y',(Y(J),J=1,P)
      PRINT*, '  '
 1235 IFLAGG=1
C --------------------SOLVE DIFFERENTIAL EQUATIONS---------------------
      CALL ODE (CLPASS,NN,X,T,TOUT,RELERR,ABSERR,IFLAGG,WORK,IWORK)
      IF (IFLAGG.NE.2) GO TO 1290
C --------------------FORM NEW OUTPUT VECTOR--------------------------
      CALL YOUT (X,Y,W,C,D)
      IF (ISKIP.NE.1) GO TO 1240
      IF (JJ.LT.NTT) PRINT*, '           T=',T
 1240 CONTINUE
C --------------------EXIT COMPUTATIONS--------------------------------
      DO 1245 I=1,P
 1245 Z(I)=Z(I)+SAMT*E(I)
 1250 CONTINUE
      IF (ISKIP.NE.1) GO TO 1255
      PRINT*, '           T=',T
      IF (ILIM.NE.0) PRINT 4065, '*',(U(K),K=1,M)
      PRINT 4065, 'U',(U(K),K=1,M)
      PRINT 4065, 'Y',(Y(J),J=1,P)
 1255 IFLAG(26)=1
C --------------------PRINT WHEN CONTROL LIMITS ARE SET-----------------
      IF (ILIM.NE.1) GO TO 1265
      PRINT '(/A)', ' ENTER "0" TO SKIP CONTROL LIMITS INFORMATION'
      PRINT*, 'ENTER "1" TO OBTAIN THIS DATA...>'
      READ*, ISKIP
      PRINT*, '  '
      IF (ISKIP.NE.1) GO TO 1265
      DO 1260 I=1,M
      PRINT*, '  '
      PRINT*, 'CONTROL #',I
      PRINT*, '     HIT THE MINIMUM STOP ',IMIN(I),' TIMES IN THE ',NT,'
     * SAMPLING POINTS'
      PRINT*, '     HIT THE MAXIMUM STOP ',IMAX(I),' TIMES IN THE ',NT,'
     * SAMPLING POINTS'
 1260 CONTINUE
C --------------------SET NT VALUE EQUAL TO PACKING SIZE---------------
 1265 NT=INT(NT/IPACK)
      PRINT*, '  '
      PRINT*, 'SIMULATION FOR SAMPLING TIME #',IG,' COMPLETE'
      PRINT*, '  '
 1285 CONTINUE
      GO TO 8007
C ***********************************************************************
C  OPTION #27
 2027 PRINT*, 'THIS OPTION ENTER INPUT CONTROL LIMITATIONS'
      PRINT '(/A)', ' ENTER LIMITS ON CONTROL INPUTS BY GIVING THE INPUT
     * NUMBER'
      PRINT*, '    FOLLOWED BY ITS MIN AND MAX VALUES'
      PRINT*, 'ENTER "0" IF NO LIMITS ARE IMPOSED'
      PRINT*, 'ENTER "99" WHEN ALL LIMITS HAVE BEEN ENTERED'
 1286 PRINT*, 'CONTROL INPUT > #'
      READ*, ILIM
      IF (ILIM.EQ.99) THEN
          PRINT*, 'ALL LIMITS HAVE BEEN ENTERED'
          IFLAG(27)=1
          GO TO 8007
C --------------------ELIMINATE CONTROL LIMITS-------------------------
      ELSEIF (ILIM.EQ.0) THEN
          DO 1287 I=1,10
          UMIN(I)=-1.E10
 1287     UMAX(I)=1.E10
          IFLAG(28)=0
      PRINT*, 'LIMITATIONS ARE ELIMINATED'
          GO TO 8007
C --------------------SET CONTROL LIMITS------------------------------
      ELSE
          PRINT*, 'MINIMUM VALUE >'
          READ*, UMIN(ILIM)
          PRINT*, 'MAXIMUM VALUE >'
          READ*, UMAX(ILIM)
          PRINT*, '  '
          GO TO 1286
      ENDIF
      GO TO 8007
C ***********************************************************************
C  OPTION #28
 2028 IERR=44
      NOPT=NOPT+20
      GO TO 9007
C ***********************************************************************
C  OPTION #29
 2029 PRINT*, 'THIS OPTION READS SIMULATION PARAMETERS FROM A LOCAL DATA
     * FILE'
      IF (IFLAG(IERR).EQ.0) GO TO 9007
 1288 PRINT '(/A)', ' ENTER LOCAL FILE NAME THAT HOLDS SIMULATION DATA'
      PRINT*, '                          >'
      READ '(A)', LFN
      PRINT*, 'LOCAL FILE NAME IS >',LFN
      CALL ANSWER (*1288,*8007)
      DATC=40
      OPEN (DATC,FILE=LFN)
      REWIND DATC
      READ (DATC,*) (XO(I),I=1,N)
      READ (DATC,*) (ZO(I),I=1,P)
      READ (DATC,*) (V(I),I=1,P)
      READ (DATC,*) NG,(GN(I),I=1,NG)
      READ (DATC,*) TT
      READ (DATC,*) ST
      PRINT*, 'DATA COPY COMPLETE FOR OPTIONS'
      PRINT*, '#21, #22, #23, #24, #25'
      DO 1289 IERR=21,25
 1289 IFLAG(IERR)=1
      CLOSE (DATC,STATUS='KEEP')
      GO TO 8007
C *********************************************************************
C *********************************************************************
C  FORMAT STATEMENTS
C *********************************************************************
 4060 FORMAT (3X,10(1E10.4,3X))
 4065 FORMAT (1X,A,1X,10(1E10.4,3X))
C ---------------------------------------------------------------------
 1290 PRINT*, 'IFLAGG HAS BEEN SET AT ',IFLAGG
      PRINT*, 'REFER TO O.D.E MANUAL FOR ERROR CODE LISTING'
 8007 IERR=0
 9007 CONTINUE
      END
C *********************************************************************
C *********************************************************************
C THIS SUBROUTINE FORMS THE DIFFERENTIAL EQUATIONS
      SUBROUTINE CLPASS(T,X,XDOT)
      DIMENSION X(50),XDOT(40)
      INTEGER P,PD
      COMMON /B  2 / N,M,P,ND,MD,PD
      COMMON /B  4 / A(10,10),B(10,10),C(10,10)
      COMMON /B  7A/ NA(10),AA(10,2,2),BA(10,2),CA(10,2)
      COMMON /B  7S/ NS(10),AS(10,2,2),BS(10,2),CS(10,2)
      COMMON /B 12 / W(10),Q(10),U(10),YM(10)
      L=N
      DO 1300 I=1,P
      S=0.
      DO 1295 J=1,NA(I)
      L=L+1
 1295 S=S+CA(I,J)*X(L)
 1300 W(I)=S
      DO 1310 I=1,P
      S=0.
      DO 1305 J=1,NS(I)
      L=L+1
 1305 S=S+CS(I,J)*X(L)
 1310 D(I)=S
      DO 1325 I=1,N
      S=0.
      DO 1315 J=1,N
 1315 S=S+A(I,J)*X(J)
      DO 1320 J=1,P
 1320 S=S+B(I,J)*W(J)
 1325 XDOT(I)=S
      L=N
      LI=N
      DO 1340 I=1,P
      DO 1335 J=1,NA(I)
      L=L+1
      S=BA(I,J)*U(I)
      DO 1330 K=1,NA(I)
 1330 S=S+AA(I,J,K)*X(LI+K)
 1335 XDOT(L)=S
 1340 LI=LI+NA(I)
      DO 1355 I=1,P
      DO 1350 J=1,NS(I)
      L=L+1
      S=BS(I,J)*YM(I)
      DO 1345 K=1,NS(I)
 1345 S=S+AS(I,J,K)*X(LI+K)
 1350 XDOT(L)=S
 1355 LI=LI+NS(I)
      RETURN
      END
C **********************************************************************
C **********************************************************************
C THIS SUBROUTINE CALCULATES THE OUTPUT VALUES
      SUBROUTINE YOUT (X,Y,W,C,D)
      INTEGER P,PD
      COMMON /B  2 / N,M,P,ND,MD,PD
      DIMENSION X(50),Y(10),W(10),C(10,10),D(10,10)
      S=0.
      DO 1375 I=1,P
      DO 1365 J=1,N
 1365 S=S+C(I,J)*X(J)
      DO 1370 K=1,M
 1370 S=S+D(I,K)*W(K)
      Y(I)=S
      S=0.
 1375 CONTINUE
      RETURN
      END
C **********************************************************************
C **********************************************************************
C  OPTIONS #30 THRU #39
C **********************************************************************
```

```
C **********************************************************************
      OVERLAY (10,0)
      PROGRAM OPTPLT
      CHARACTER CHOICE*6
      COMMON /B  1 / NOPT,IERR,IPLANT,IFLAG(40)
      COMMON /B 12B/ ST,TT
      COMMON /B 13 / N,Y(100,11,2)
      COMMON /B 13A/ U(100,11,2)
      COMMON /B 14 / ICODE,NUM,ICOLMN(4),PLMAT(102,4)
      COMMON /B 14A/ CHOICE
      COMMON /B 15 / ICLM,NPAIRS,NDEPTH,IDEPTH(2)
      COMMON /B 16 / IENTRY,IFLTCN,LINES
      COMMON /B 17 / IPLOT
      DIMENSION XAXIS(102),YAXIS(102)
      DATA XAXIS,YAXIS/204*0./
C ----------------------------------------------------------------------
      IF (NOPT.EQ.33) THEN
          IF (IFLAG(31).EQ.0.AND.IFLAG(32).EQ.0) GO TO 1525
      ENDIF
      IF (NOPT.EQ.36) THEN
          IF (IFLAG(34).EQ.0.AND.IFLAG(35).EQ.0) GO TO 1525
      ENDIF
      IERR=26
      IENTRY=IENTRY+1
      IFLAG(NOPT)=1
      NOPT=NOPT-30
      GO TO (2031,2031,2031,2031,2031,2031,2037,2038,2039) NOPT
C **********************************************************************
C   OPTION  #30
      PRINT*, 'PLOTTING OPTIONS:'
      PRINT*, '    30.  LIST OPTIONS 30 THRU 39'
      PRINT*, '    31.  QUICK SKETCH AT USER''S TERMINAL'
      PRINT*, '    32.  QUICK SKETCH--SHORT VERSION'
      PRINT*, '    33.  QUICK SKETCH--RETAINING SAME PLOTTING CHOICES'
      PRINT*, '    34.  CALCOMP PLOT'
      PRINT*, '    35.  CALCOMP PLOT--SHORT VERSION'
      PRINT*, '    36.  CALCOMP PLOT--RETAINING SAME PLOTTING CHOICES'
      PRINT*, '    37.  OPTION RESERVED'
      PRINT*, '    38.  OPTION RESERVED'
      PRINT*, '    39.  OPTION RESERVED'
      GO TO 8010
C **********************************************************************
C   OPTIONS #31 THRU #36
 2031 IF (NOPT.GE.4.AND.NOPT.LE.6) GO TO 1400
      PRINT '(A/)', ' THIS OPTION PRODUCES A PLOT AT YOUR TERMINAL'
      IF (IFLAG(IERR).EQ.0) GO TO 9010
      GO TO 1405
 1400 PRINT '(A/)', ' THIS OPTION PRODUCES A CALCOMP PLOT'
      IF (IFLAG(IERR).EQ.0) GO TO 9010
C --------------------CHECK FOR MULTIPLE FLIGHT CONDITIONS--------------
```

```
 1405 IF (ICODE.NE.3) GO TO 1410
      PRINT*, '  '
      PRINT*, '-----------------------------------------------------
     *----------'
      PRINT*, 'YOU CHOSE TO OBTAIN A MULTIPLE PLOT FOR DIFFERENT FLIGHT
     *CONDITIONS'
      PRINT*, '              ON THE LAST ENTRY TO THIS OPTION'
      IF (IFLTCN.EQ.1) THEN
          PRINT*, '   THE PLOT DATA IS NOW COMPLETE WITH THIS SIMULATION
     * DATA ENTRY'
          PRINT*, '-------------------------------------------------
     *--------------'
          PRINT*, '  '
      ENDIF
      GO TO 1460
C --------------------SET DATA FOR SHORT VERSION PLOT-------------------
 1410 IF (NOPT.EQ.3.OR.NOPT.EQ.6) GO TO 1460
      IF (NOPT.EQ.2.OR.NOPT.EQ.5) THEN
          PRINT*, 'ENTER SHORT VERSION CHOICE...1,2,3,4...>'
          READ*, ICODE
          NPAIRS=1
          IF (ICODE.NE.4) NDEPTH=1
          IF (ICODE.EQ.1) THEN
              CHOICE='   ONE'
              NUM=NPAIRS*2
          ELSE
              PRINT*, 'ENTER...INPUT,OUTPUT...>'
              READ '(A)', CHOICE
              IF (CHOICE.EQ.'INPUT ') CHOICE=' INPUT'
          ENDIF
          IF (ICODE.EQ.1.AND.NOPT.EQ.5) THEN
              PRINT*, 'ENTER NUMBER OF PAIRS...>'
              READ*, NPAIRS
          ENDIF
          PRINT*, 'ENTER PROPER PLOT DATA...>'
          IF (ICODE.EQ.1) THEN
              NUM=NPAIRS*2
              ICLM=NPAIRS
              READ*, IDEPTH(1),(ICOLMN(I),I=1,NUM)
          ELSEIF (ICODE.EQ.2) THEN
              READ*, IDEPTH(1),NUM,(ICOLMN(I),I=1,NUM)
              ICLM=NUM
          ELSEIF (ICODE.EQ.3) THEN
              NUM=1
              ICLM=NUM
              READ*, IFLTCN,IDEPTH(1),ICOLMN(1)
              LINES=IFLTCN
          ELSEIF (ICODE.EQ.4) THEN
              NUM=1
              ICLM=NUM
```

```
            READ*, NDEPTH,(IDEPTH(I),I=1,NDEPTH),ICOLMN(1)
            GO TO 1420
         ENDIF
         GO TO 1460
      ENDIF
C --------------------SET ICODE----------------------------------------
      PRINT*, 'PLEASE CHOOSE ONE OF THE FOLLOWING:'
      PRINT*, '  '
      PRINT*, 'FOR A SINGLE SAMPLING TIME'
      PRINT*, '   1...A PLOT OF UP TO 2 INPUT AND OUTPUT PAIRS'
      PRINT*, '   2...A PLOT OF UP TO 4 INPUTS OR OUTPUTS'
      PRINT*, '   3...A PLOT OF UP TO 4 DIFFERENT SIMULATIONS'
      PRINT*, '         (FOR ANY SINGLE INPUT OR OUTPUT)'
      PRINT*, '  '
      PRINT*, 'OR FOR UP TO 4 DIFFERENT SAMPLING TIMES'
      PRINT*, '   4...A PLOT OF ANY SINGLE INPUT OR OUTPUT'
      PRINT*, '  '
      PRINT*, 'ENTER CHOICE DESIRED >'
      READ*, ICODE
 1420 IF (ICODE.EQ.4) THEN
         IF (Y(N,1,1).NE.U(N,1,1)) THEN
            PRINT '(/A)', ' STEP SIZE OR RUN TIME ARE NOT SET CORRECTL
     *Y'
            PRINT*, 'TO PRODUCE EQUAL NUMBER OF TIME INTERVALS'
         GO TO 8010
         ENDIF
      IF (NOPT.EQ.2.OR.NOPT.EQ.5) GO TO 1460
      ENDIF
      PRINT*, '  '
      GO TO (1425,1430,1435,1440) ICODE
C --------------------SET NPAIRS (1), SET IFLTCN & LINES (3)------------
 1425 PRINT*, 'CHOICE #1...YOU''VE CHOSEN TO PLOT INPUT AND OUTPUT PAIRS
     *'
      IF (NOPT.EQ.1) THEN
         NPAIRS=1
      ELSE
         PRINT*, '  '
         PRINT*, 'ENTER THE NUMBER OF PAIRS TO BE PLOTTED >'
         READ*, NPAIRS
      ENDIF
      GO TO 1445
 1430 PRINT*, 'CHOICE #2...YOU''VE CHOSEN TO PLOT INPUTS OR OUTPUTS'
      GO TO 1445
 1435 PRINT*, 'CHOICE #3...YOU''VE CHOSEN TO PLOT 1 INPUT OR 1 OUTPUT'
      PRINT*, '        FOR VARIOUS SIMULATIONS'
      PRINT*, '  '
      PRINT*, 'ENTER THE NUMBER OF SIMULATIONS TO BE PLOTTED >'
      READ*, IFLTCN
      LINES=IFLTCN
      GO TO 1445
```

```
  1440 PRINT*, 'CHOICE #4...YOU''VE CHOSEN TO PLOT 1 INPUT OR 1 OUTPUT'
       PRINT*, '              FOR VARIOUS SAMPLING TIMES'
       PRINT*, '  '
C --------------------SET NDEPTH AND IDEPTH(ARRAY)----------------------
  1445 IF (ICODE.EQ.4) THEN
           PRINT*, 'ENTER THE NUMBER OF DIFFERENT SAMPLING TIMES TO BE PL
      *OTTED >'
           READ*, NDEPTH
           PRINT*, '  '
       ELSE
           NDEPTH=1
           PRINT*, '  '
           PRINT*, 'YOU MIGHT HAVE SELECTED TO RUN SEVERAL SAMPLING TIMES
      *...'
       ENDIF
       PRINT*, 'DESIGNATE THESE SAMPLING TIMES AS...1,2,3, ETC...'
       PRINT*, '     ENTER THE ',NDEPTH,' SAMPLING TIME(S) OF INTEREST >'
       READ*, (IDEPTH(I),I=1,NDEPTH)
       PRINT*, '  '
C --------------------SET CHOICE, SET NUM, PRINT MESSAGES (1)-----------
       IF (ICODE.EQ.1) THEN
           CHOICE='   ONE'
           NUM=NPAIRS*2
       ENDIF
       IF (ICODE.EQ.1) THEN
           IF (NOPT.EQ.1) THEN
               PRINT*, '  '
               PRINT*, '-------------------------------NOTE--------------
      *------------------'
               PRINT*, '      YOU CAN ONLY PRINT 1 INPUT VS. 1 OUTPUT WHE
      *N REQUESTING'
               PRINT*, '                         A PLOT AT YOUR TERMINAL'
           ENDIF
  1450     PRINT*, '-----------------------------------------------------
      *--------------'
           PRINT*, '      COLUMN ENTRIES SHOULD BE INPUT #S FIRST, THEN O
      *UTPUT #S'
           PRINT*, '-----------------------------------------------------
      *--------------'
           PRINT*, '  '
           GO TO 1455
       ELSE
           PRINT*, 'ENTER...INPUT OR OUTPUT...FOR YOUR PLOT >'
           READ '(A)', CHOICE
           IF (CHOICE.EQ.'INPUT ') CHOICE=' INPUT'
           NUM=1
       ENDIF
       IF (ICODE.EQ.2) THEN
       PRINT*, '  '
           PRINT*, 'HOW MANY ',CHOICE,'S DO YOU WANT TO PLOT >'
```

```
              READ*, NUM
         ENDIF
C --------------------SET ICOLMN(ARRAY)--------------------------------
  1455   PRINT*, ' '
         PRINT*, 'WHICH ',CHOICE,'(S) DO YOU WANT TO PLOT...'
         PRINT*, '     ENTER THE ',NUM,' CORRESPONDING COLUMN NUMBER(S) >'
         READ*, (ICOLMN(I),I=1,NUM)
         PRINT*, ' '
C --------------------SET ICLM------------------------------------------
         IF (ICODE.EQ.1) THEN
           ICLM=NPAIRS
         ELSE
           ICLM=NUM
         ENDIF
C --------------------ENTRY POINT FOR MULTIPLE SIMULATIONS--------------
  1460   CONTINUE
C --------------------SET PLMAT(MATRIX)---------------------------------
         IF (ICODE.EQ.4) GO TO 1495
         IF (CHOICE.EQ.'OUTPUT') GO TO 1480
         DO 1475 I=1,N
         DO 1475 J=1,ICLM
         IF (ICODE.EQ.3) GO TO 1465
C - - - - - - - - - - INPUT(1,2) - - - - - - - - - - - - - - - - - - - -
         PLMAT(I,J)=U(I,(ICOLMN(J)+1),IDEPTH(NDEPTH))
         GO TO 1470
C - - - - - - - - - - INPUT(3) - - - - - - - - - - - - - - - - - - - - -
  1465   PLMAT(I,IENTRY)=U(I,ICOLMN(J)+1,IDEPTH(NDEPTH))
  1470   IF (ICODE.NE.1) GO TO 1475
C - - - - - - - - - - OUTPUT(1) - - - - - - - - - - - - - - - - - - - -
         PLMAT(I,J+ICLM)=Y(I,(ICOLMN(J+ICLM)+1),IDEPTH(NDEPTH))
  1475   CONTINUE
         GO TO 1515
  1480   DO 1490 I=1,N
         DO 1490 J=1,ICLM
         IF (ICODE.EQ.3) GO TO 1485
C - - - - - - - - - - OUTPUT(2) - - - - - - - - - - - - - - - - - - - -
         PLMAT(I,J)=Y(I,ICOLMN(J)+1,IDEPTH(NDEPTH))
         GO TO 1490
C - - - - - - - - - - OUTPUT(3) - - - - - - - - - - - - - - - - - - - -
  1485   PLMAT(I,IENTRY)=Y(I,ICOLMN(J)+1,IDEPTH(NDEPTH))
  1490   CONTINUE
         GO TO 1515
  1495   IF (CHOICE.EQ.'OUTPUT') GO TO 1505
         DO 1500 I=1,N
         DO 1500 K=1,NDEPTH
C - - - - - - - - - - INPUT(4) - - - - - - - - - - - - - - - - - - - - -
  1500   PLMAT(I,K)=U(I,ICOLMN(ICLM)+1,IDEPTH(K))
         GO TO 1515
  1505   DO 1510 I=1,N
         DO 1510 K=1,NDEPTH
```

```
C - - - - - - - - - - - OUTPUT(4) - - - - - - - - - - - - - - - - - - -
 1510 PLMAT(I,K)=Y(I,ICOLMN(ICLM)+1,IDEPTH(K))
C ---------------------------------------------------------------------
 1515 IF (ICODE.NE.3) IFLTCN=1
      IFLTCN=IFLTCN-1
      IF (IFLTCN.NE.0) GO TO 1520
      IF (ICODE.EQ.3) ICLM=LINES
      IF (ICODE.EQ.4) ICLM=NDEPTH
      GO TO 8010
 1520 PRINT*, '-----------------------------------------------------------
     *----------'
      PRINT*, '          THE ',CHOICE,' DATA FOR THIS SIMULATION HAS BEEN
     *SAVED'
      PRINT*, '                THE PROGRAM NOW EXITS THE PLOT ROUTINE'
      PRINT*, '     THE NEXT ',IFLTCN,' CALL(S) TO OPTIONS #31 THRU #36
     *WILL CONTINUE'
      PRINT*, '                     TO ADD INFORMATION TO THIS DATA'
      PRINT*, '----------------------------WARNING---------------------
     *----------'
      PRINT*, '        FOR A PROPER PLOT BETWEEN DIFFERENT SIMULATIONS
     *'
      PRINT*, '      THE SAME SIMULATION TIME AND STEP SIZE MUST BE US
     *ED'
      PRINT*, '       SIMULATION TIME WAS ',TT,' AND THE STEP SIZE WAS
     * ',ST
      PRINT*, '-----------------------------------------------------------
     *----------'
      GO TO 8010
C *********************************************************************
C  OPTION #37
 2037 IERR=44
      NOPT=NOPT+30
      GO TO 9010
C *********************************************************************
C  OPTION #38
 2038 IERR=44
      NOPT=NOPT+30
      GO TO 9010
C *********************************************************************
C  OPTION #39
 2039 IERR=44
      NOPT=NOPT+30
      GO TO 9010
C ---------------------------------------------------------------------
 8010 IERR=0
      GO TO 9010
 1525 IERR=30
 9010 CONTINUE
      END
C *********************************************************************
```

```
C *********************************************************************
C  OVERLAY FOR TERMINAL PLOT
C *********************************************************************
C *********************************************************************
      OVERLAY (11,0)
      PROGRAM OPT31
      CHARACTER STRING*10,CODICE*6,KHAR
      COMMON /B 13 / N,Y(102,11,2)
      COMMON /B 14 / ICODE,NUM,ICOLMN(4),PLMAT(102,4)
      COMMON /B 14A/ CODICE
      COMMON /B 15 / ICLM,NPAIRS,NDEPTH,IDEPTH(2)
      COMMON /B 16 / IENTRY,IFLTCN,LINES
      COMMON /SCALIT/ NX,NY,NXT,NYT,LINPIN,IGRID
      DIMENSION XAXIS(102),YAXIS(102)
      DATA STRING/'XO*=$%&#'/
      DATA XAXIS,YAXIS/204*0./
C --------------------SET PLOTIT PARAMETERS----------------------------
      PRINT*, 'FOR NO GRID ON PLOT ENTER "0", FOR A GRID ENTER "1" >'
      READ*, IGRID
      NX=60
      NY=20
      NXT=10
      NYT=10
      LINPIN=8
C --------------------FORM TIME AXIS VECTOR----------------------------
      DO 1600 I=1,N
 1600 XAXIS(I)=Y(I,1,IDEPTH(NDEPTH))
C --------------------FIND MIN & MAX VALUES FOR SCALING----------------
      XMAX=XAXIS(N)
      YMIN=PLMAT(1,1)
      YMAX=PLMAT(1,1)
      IF (ICODE.EQ.1) ICLM=NUM
      DO 1605 I=1,N
      DO 1605 J=1,ICLM
      YMIN=MIN(YMIN,PLMAT(I,J))
 1605 YMAX=MAX(YMAX,PLMAT(I,J))
      ADD=(YMAX-YMIN)/20.
      YMIN=YMIN-ADD
      YMAX=YMAX+ADD
C --------------------LOOP FOR PLOTTING EACH COLUMN OF DATA------------
      IF (ICODE.EQ.1) ICLM=NPAIRS
      DO 1645 J=1,ICLM
C --------------------FOR EACH YAXIS VECTOR OF DATA--------------------
      DO 1610 I=1,N
 1610 YAXIS(I)=PLMAT(I,J)
      IF (ICODE.EQ.1) THEN
          KHAR='X'
      ELSE
           KHAR=STRING(J:J)
      ENDIF
```

AD-A118 134   AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 1/3
DESIGN AND ANALYSIS OF A MULTIVARIABLE, DIGITAL CONTROLLER FOR --ETC(U)
DEC 81 D S PORTER
UNCLASSIFIED   AFIT/GE/EE/81D-48   NL

3 OF 3
AD
A118 134

END
DATE
FILMED
09-82
DTIC

```
      INDEX=-1
      IF (J.EQ.ICLM.AND.ICODE.NE.1) INDEX=-2
      CALL PLOTIT (XAXIS,YAXIS,N,KHAR,0.,XMAX,YMIN,YMAX,11,INDEX)
      IF (ICODE.NE.1) GO TO 1620
      DO 1615 I=1,N
 1615 YAXIS(I)=PLMAT(I,J+NPAIRS)
      KHAR='O'
      INDEX=-2
      CALL PLOTIT (XAXIS,YAXIS,N,KHAR,0.,XMAX,YMIN,YMAX,11,INDEX)
 1620 IF (INDEX.NE.-2) GO TO 1645
C --------------------EXIT TO PRINT STATEMENTS WHEN PLOT IS DISPLAYED---
      IF (ICODE.EQ.1) GO TO 1640
      DO 1635 I=1,ICLM
      IF (ICODE.EQ.3) GO TO 1625
      IF (ICODE.EQ.4) GO TO 1630
      PRINT*, 'CURVE ',STRING(I:I),' ABOVE IS ',CHOICE,' ', ICOLMN(I)
      GO TO 1635
 1625 PRINT*, 'CURVE ',STRING(I:I),' ABOVE IS ',CHOICE,' ',ICOLMN(1),' F
     *OR FLIGHT CONDITION ',I
      GO TO 1635
 1630 PRINT*, 'CURVE ',STRING(I:I),' ABOVE IS ',CHOICE,' ',ICOLMN(1),' F
     *OR SAMPLING TIME ',I
 1635 CONTINUE
      GO TO 1645
 1640 PRINT*, 'CURVE X REPRESENTS INPUT ',ICOLMN(1)
      PRINT*, 'CURVE O REPRESENTS OUTPUT ', ICOLMN(2)
 1645 CONTINUE
      ICODE=0
      IENTRY=0
C ----------------------------------------------------------------------
      END
C **********************************************************************
C **********************************************************************
C  OVERLAY FOR CALCOMP PLOT
C **********************************************************************
C **********************************************************************
      OVERLAY (12,0)
      PROGRAM OPT34
      CHARACTER YTITLE*30,CHOICE*6,XTITLE*20,LFN*60
      COMMON /B  1 / NOPT,IERR,IPLANT,IFLAG(40)
      COMMON /B 13 / N,Y(100,11,2)
      COMMON /B 14 / ICODE,NUM,ICOLMN(4),PLMAT(102,4)
      COMMON /B 14A/ CHOICE
      COMMON /B 15 / ICLM,NPAIRS,NDEPTH,IDEPTH(2)
      COMMON /B 16 / IENTRY,IFLTCN,LINES
      COMMON /B 17 / IPLOT
      DIMENSION XAXIS(102),YAXIS(102)
      DIMENSION ABSC(4),ORD(4)
      DATA XAXIS,YAXIS/204*0./
      DATA ABSC,ORD/8*0./
      DATA XTITLE/'    TIME, SECONDS   '/
C --------------------COUNT # OF PLOTS IN FILE------------------------
      IPLOT=IPLOT+1
C --------------------CREATE TIME AXIS--------------------------------
      DO 1700 I=1,N
 1700 XAXIS(I)=Y(I,1,IDEPTH(NDEPTH))
C --------------------FIND MIN & MAX VALUES FOR SCALING---------------
      YMIN=PLMAT(1,1)
      YMAX=PLMAT(1,1)
      IF (ICODE.EQ.1) ICLM=NUM
      DO 1705 I=1,N
      DO 1705 J=1,ICLM
      YMIN=MIN(YMIN,PLMAT(I,J))
 1705 YMAX=MAX(YMAX,PLMAT(I,J))
C --------------------CREATE YAXIS FOR SCALING------------------------
      YAXIS(1)=YMIN
      YAXIS(2)=YMAX
      DO 1706 I=3,N
 1706 YAXIS(I)=PLMAT(I,2)
C --------------------OBTAIN PLOT TITLES------------------------------
 1710 PRINT*, 'ENTER TITLE FOR Y-AXIS OF PLOT (MAX OF 30 CHARACTERS)'
      PRINT*, '                                            <'
      PRINT '(A)', '+             >'
      READ '(A)', YTITLE
      PRINT*, 'TITLE IS:    ',YTITLE
      CALL ANSWER (*1710,*1730)
 1712 PRINT*, 'ENTER MAIN PLOT TITLE (MAX 60 CHARACTERS)'
      PRINT*, '
     *          <'
      PRINT '(A)', '+             >'
      READ '(A)', LFN
      PRINT*, 'MAIN TITLE:  ',LFN
      CALL ANSWER (*1712,*1730)
C ---------------------BEGIN CALCOMP PLOTTING CALLS-------------------
      CALL PLOTS (ABM,BAM,MMAB)
C --------------------SET PLOT SIZE-----------------------------------
      PRINT*, 'ENTER PLOT SIZE FACTOR...>'
      READ*, FACTR
      PRINT*, '  '
      CALL FACTOR (FACTR)
C --------------------SCALE PLOT--------------------------------------
      CALL SCALE (XAXIS,8.,N,1)
      CALL SCALE (YAXIS,5.,N,1)
      DO 1715 J=1,ICLM
      PLMAT(N+1,J)=YAXIS(N+1)
 1715 PLMAT(N+2,J)=YAXIS(N+2)
      ABSC(3)=XAXIS(N+1)
      ABSC(4)=XAXIS(N+2)
      ORD(3)=YAXIS(N+1)
      ORD(4)=YAXIS(N+2)
```

```
C --------------------CREATE ORIGIN-----------------------------------
      CALL PLOT (2.,2.,-3)
C --------------------PLOT X, Y & ORIGIN AXIS'S-----------------------
      CALL AXIS (0.,0.,XTITLE,-20,8.,0.,XAXIS(N+1),XAXIS(N+2))
      CALL AXIS (0.,0.,YTITLE,30,5.,90.,YAXIS(N+1),YAXIS(N+2))
      IF (YMIN.LT.0.) THEN
         ABSC(2)=XAXIS(N)
         CALL LINE (ABSC,ORD,2,1,2,17)
      ENDIF
C --------------------PLOT EACH CURVE, SYMBOL & BOX--------------------
      IF (ICODE.EQ.1) ICLM=NPAIRS
      DO 1725 J=1,ICLM
      DO 1720 I=1,N+2
 1720 YAXIS(I)=PLMAT(I,J)
      XSYM=((XAXIS(N)-XAXIS(N+1))/XAXIS(N+2))+.25
      YSYM=(YAXIS(N)-YAXIS(N+1))/YAXIS(N+2)
      CALL FLINE (XAXIS,YAXIS,-N,1,1,72)
      CALL SYMBOL (XSYM,YSYM,.105,54+ICOLMN(J),0.,-1)
      IF (ICODE.NE.1) GO TO 1725
      DO 1722 I=1,N+2
 1722 YAXIS(I)=PLMAT(I,J+NPAIRS)
      YSYM=(YAXIS(N)-YAXIS(N+1))/YAXIS(N+2)
      CALL FLINE (XAXIS,YAXIS,-N,1,1,72)
      CALL SYMBOL (XSYM,YSYM,.105,54+ICOLMN(J),0.,-1)
 1725 CONTINUE
      CALL RECT (-0.5,-0.5,6.,9.,0.,3)
C --------------------TERMINATE PLOTTING CALLS------------------------
      CALL SYMBOL (0.,-.8,.14,LFN,0.,60)
      CALL PLOTE (BLK)
      NOPT=NOPT+30
      PRINT*, '-------------------------------------------------------
     *---------------'
      PRINT*, 'LOCAL FILE "PLOT" CONTAINS THE CALCOMP DATA FOR THIS ENTR
     *Y TO OPTION #',NOPT
      PRINT*, '                YOU HAVE GENERATED A TOTAL OF ',IPLOT,' P
     *LOT(S)'
      IF (IPLOT.GE.5) THEN
      PRINT*, '            SINCE 5 CALCOMP PLOTS ON ONE FILE ARE THE LIM
     *IT'
         PRINT*, '    PLEASE USE OPTION #99 NEXT, AND ROUTE "PLOT" TO T
     *O THE PRINTER'
      PRINT*, '-------------------------------------------------------
     *---------------'
      ELSE
         PRINT*, '          BE SURE TO ROUTE "PLOT" TO THE PRINTER BEFO
     *RE LOGOUT'
      PRINT*, '-------------------------------------------------------
     *---------------'
      ENDIF
C -------------------------------------------------------------------
```

```
      IERR=0
      IENTRY=0
      ICODE=0
      GO TO 9012
 1730 IPLOT=IPLOT-1
 9012 CONTINUE
C ---------------------------------------------------------------------
      END
C *********************************************************************
C *********************************************************************
C  OVERLAY FOR ERROR STATEMENTS
C *********************************************************************
C *********************************************************************
      OVERLAY (13,0)
      PROGRAM ERROR
      COMMON /B  1 / NOPT,IERR,IPLANT,IFLAG(40)
      PRINT*, '  '
      GO TO (3001,3002,3003,3004,3005,3006,3007,3008,3009,
     *3010,3011,3012,3013,3014,3015,3016,3017,3018,3019,
     *3020,3021,3022,3023,3024,3025,3026,3027,3028,3029,
     *3030,3031,3032,3033,3034,3035,3036,3037,3038,3039,
     *3040,3041,3042,3043,3044)IERR
C --------------------OPTION ERRORS------------------------------------
 3001 PRINT*, 'G(0) MATRIX MISSING...SEE OPTION #1'
      GO TO 8013
 3002 PRINT*, '# OF STATES, INPUTS & OUTPUTS MISSING...SEE OPTION #2'
      GO TO 8013
 3003 PRINT*, 'A, B, C & D MATRICES MISSING...SEE OPTION #3'
      GO TO 8013
 3004 PRINT*, 'ACTUATOR STATE EQUATIONS MISSING...SEE OPTION #4'
      GO TO 8013
 3005 PRINT*, 'SENSOR STATE EQUATIONS MISSING...SEE OPTION #5'
      GO TO 8013
 3006 PRINT*, 'ERROR #6 STATEMENT'
      GO TO 8013
 3007 PRINT*, 'ERROR #7 STATEMENT'
      GO TO 8013
 3008 PRINT*, 'LOCAL FILE HOLDS STATE, ACTUATOR & SENSOR INFO...SEE OPTI
     *ON #9'
      GO TO 8013
 3009 PRINT*, 'LOCAL FILE HOLDS G(0) INFO...SEE OPTION #8'
      GO TO 8013
 3010 PRINT*, 'ERROR #10 STATEMENT'
      GO TO 8013
 3011 PRINT*, 'ALPHA MISSING...SEE OPTION #11'
      GO TO 8013
 3012 PRINT*, 'SIGMA WEIGHTING MATRIX MISSING...SEE OPTION #12'
      GO TO 8013
 3013 PRINT*, 'SIGMA MATRIX MULTIPLIER MISSING...SEE OPTION #13'
      GO TO 8013
 3014 PRINT*, 'K0 & K1 MATRICES MISSING...SEE OPTION #14'
      GO TO 8013
 3015 PRINT*, 'ERROR #15 STATEMENT'
      GO TO 8013
 3016 PRINT*, 'ERROR #16 STATEMENT'
      GO TO 8013
 3017 PRINT*, 'ERROR #17 STATEMENT'
      GO TO 8013
 3018 PRINT*, 'MEASUREMENT MATRIX MISSING...SEE OPTION #18'
      GO TO 8013
 3019 PRINT*, 'ERROR #19 STATEMENT'
      GO TO 8013
 3020 PRINT*, 'ERROR #20 STATEMENT'
      GO TO 8013
 3021 PRINT*, 'STATE AND INTEGRATOR INITIAL CONDITIONS MISSING...SEE OPT
     *ION #21'
      GO TO 8013
 3022 PRINT*, 'COMMAND VECTOR MISSING...SEE OPTION #22'
      GO TO 8013
 3023 PRINT*, 'SAMPLING TIMES MISSING...SEE OPTION #23'
      GO TO 8013
 3024 PRINT*, 'SIMULATION TIME MISSING...SEE OPTION #24'
      GO TO 8013
 3025 PRINT*, 'STEP SIZE MISSING...SEE OPTION #25'
      GO TO 8013
 3026 PRINT*, 'SIMULATION RUN DATA MISSING...SEE OPTION #26'
      GO TO 8013
 3027 PRINT*, 'THERE ARE NO CONTROL LIMITS SET...SEE OPTION #27'
      GO TO 8013
 3028 PRINT*, 'ERROR #28 STATEMENT'
      GO TO 8013
 3029 PRINT*, 'ERROR #29 STATEMENT'
      GO TO 8013
 3030 PRINT*, 'PLOT CHOICES HAVE NOT BEEN ENTERED'
      GO TO 8013
C ---------------------MATRIX ERRORS------------------------------------
 3031 PRINT*, 'INVERSE OF G(0) CAN''T BE FOUND'
      GO TO 8013
 3032 PRINT*, 'INVERSE OF C*B CAN''T BE FOUND'
      GO TO 8013
 3033 PRINT*, 'INVERSE OF B2 CAN''T BE FOUND'
      GO TO 8013
 3034 PRINT*, 'INVERSE OF F2 CAN''T BE FOUND'
      GO TO 8013
 3035 PRINT*, 'INVERSE OF [ (A12 * K) - A11 ] CAN''T BE FOUND'
      PRINT*, 'TRY ANOTHER K MATRIX FROM C.E.S.A.'
      GO TO 8013
 3036 PRINT*, 'INVERSE OF "A" MATRIX CAN''T BE FOUND'
      GO TO 8013
 3037 PRINT*, 'ERROR #37 STATEMENT'
```

```
      GO TO 8013
 3038 PRINT*, 'ERROR #38 STATEMENT'
      GO TO 8013
 3039 PRINT*, 'DIMENSIONING PROBLEM IN CALL TO SUBROUTINE VMULFF'
      GO TO 8013
C -------------------MISCELLANEOUS ERRORS---------------------------
 3040 PRINT*, 'OPTION ABORTED...G(0) MUST BE SQUARE'
      GO TO 8013
 3041 PRINT*, 'ERROR #41 STATEMENT'
      GO TO 8013
 3042 PRINT*, '#STATES = #INPUTS...MEASUREMENT MATRIX CAN''T BE FORMED'
      PRINT*, 'USE UNKNOWN OR REGULAR DESIGN, OR SEE INSTRUCTOR'
      GO TO 8013
 3043 PRINT*, 'THE ARE NO VALUES IN MEMORY CORE FOR OPTION #',NOPT
      GO TO 8013
 3044 PRINT*, 'OPTION #',NOPT,' NOT AVAILABLE AT THIS TIME'
C -----------------------------------------------------------------
 8013 IERR=0
      END
C *****************************************************************
C *****************************************************************
C  OPTION #99 -- MEMORY FILES
C *****************************************************************
C *****************************************************************
      OVERLAY (14,0)
      PROGRAM MEMORY
      REAL K0,K1,MM
      INTEGER P,PD,DAT1,DAT2,DAT3
      CHARACTER DMATRX,METHOD,ACT,SEN
      COMMON /B  1 / NOPT,IERR,IPLANT,IFLAG(40)
      COMMON /B  2 / N,M,P,ND,MD,PD
      COMMON /B  3 / G0(10,10)
      COMMON /B  4 / A(10,10),B(10,10),C(10,10)
      COMMON /B  5 / D(10,10)
      COMMON /B  6 / DMATRX,ACT,SEN
      COMMON /B  7A/ NA(10),AA(10,2,2),BA(10,2),CA(10,2)
      COMMON /B  7S/ NS(10),AS(10,2,2),BS(10,2),CS(10,2)
      COMMON /B  8 / ALPHA,EPSLON,GAMA(10,10)
      COMMON /B  8A/ METHOD
      COMMON /B  9 / K0(10,10),K1(10,10)
      COMMON /B 10 / NG,GN(10)
      COMMON /B 11 / MM(10,10)
      COMMON /B 12A/ X0(10),Z0(10),V(10)
      COMMON /B 12B/ ST,TT
C -------------------FORM MEMO MEMORY FILE-------------------------
      DAT1=20
      OPEN (DAT1,FILE='MEMO')
      REWIND DAT1
      IF (IPLANT.EQ.2) GO TO 1772
      DO 1750 I=2,3
```

```
1750 IF (IFLAG(I).EQ.0) IPLANT=0
     WRITE (DAT1,*) IPLANT
     WRITE (DAT1,*) N,M,P
     DO 1752 I=1,N
1752 WRITE (DAT1,*) (A(I,J),J=1,N)
     DO 1754 I=1,N
1754 WRITE (DAT1,*) (B(I,J),J=1,M)
     DO 1756 I=1,P
1756 WRITE (DAT1,*) (C(I,J),J=1,N)
     WRITE (DAT1,'(A)') DMATRX
     IF (DMATRX.EQ.'N') GO TO 1760
     DO 1758 I=1,P
1758 WRITE (DAT1,*) (D(I,J),J=1,M)
1760 WRITE (DAT1,'(A)') ACT
     IF (ACT.EQ.'N') GO TO 1766
     WRITE (DAT1,*) (NA(I),I=1,M)
     DO 1764 K=1,M
     DO 1762 I=1,NA(K)
1762 WRITE (DAT1,*) (AA(K,I,J),J=1,NA(K))
     WRITE (DAT1,*) (BA(K,I),I=1,NA(K))
1764 WRITE (DAT1,*) (CA(K,I),I=1,NA(K))
1766 WRITE (DAT1,'(A)') SEN
     IF (SEN.EQ.'N') GO TO 1776
     WRITE (DAT1,*) (NS(I),I=1,P)
     DO 1770 K=1,P
     DO 1768 I=1,NS(K)
1768 WRITE (DAT1,*) (AS(K,I,J),J=1,NS(K))
     WRITE (DAT1,*) (BS(K,I),I=1,NS(K))
1770 WRITE (DAT1,*) (CS(K,I),I=1,NS(K))
     GO TO 1776
1772 WRITE (DAT1,*) IPLANT
     WRITE (DAT1,*) M,P
     DO 1774 I=1,P
1774 WRITE (DAT1,*) (GO(I,J),J=1,M)
C --------------------FORM MEM10 MEMORY FILE---------------------------
1776 DAT2=30
     OPEN (DAT2,FILE='MEM10')
     REWIND DAT2
     WRITE (DAT2,'(A)') METHOD
     WRITE (DAT2,*) ALPHA
     WRITE (DAT2,*) (GAMA(I,I),I=1,P)
     WRITE (DAT2,*) EPSLON
     DO 1778 I=1,M
1778 WRITE (DAT2,*) (KO(I,J),J=1,P)
     IF (ALPHA.EQ.1) GO TO 1781
     DO 1780 I=1,M
1780 WRITE (DAT2,*) (K1(I,J),J=1,P)
1781 IF (METHOD.EQ.'I') THEN
          DO 1782 I=1,P
1782      WRITE (DAT2,*) (MM(I,J),J=1,N-M)
```

```
      ENDIF
C --------------------FORM MEM20 MEMORY FILE---------------------------
      DAT3=50
      OPEN (DAT3,FILE='MEM20')
      REWIND DAT3
      WRITE (DAT3,*) (XO(I),I=1,N)
      WRITE (DAT3,*) (ZO(I),I=1,P)
      WRITE (DAT3,*) (V(I),I=1,P)
      WRITE (DAT3,*) NG,(GN(I),I=1,NG)
      WRITE (DAT3,*) TT
      WRITE (DAT3,*) ST
C ---------------------------------------------------------------------
      IF (IPLANT.EQ.2) GO TO 1796
      PRINT*, 'ALL PLANT INPUT DATA HAS BEEN SAVED IN A LOCAL FILE'
      PRINT*, '     CALLED "MEMO"'
      GO TO 1798
 1796 PRINT '(/A)', ' THE G(O) MATRIX HAS BEEN SAVED IN A LOCAL FILE'
      PRINT*, '     CALLED "MEMO"'
 1798 PRINT*, ' '
      PRINT*, 'ALL DESIGN DATA HAS BEEN SAVED IN A LOCAL FILE'
      PRINT '(A/)', '      CALLED "MEM10"'
      PRINT*, 'ALL SIMULATION DATA HAS BEEN SAVED IN A LOCAL FILE'
      PRINT '(A/)', '      CALLED "MEM20"'
C ---------------------------------------------------------------------
      END
C *********************************************************************
C *********************************************************************
C  OPTIONS #100 THRU #139
C *********************************************************************
C *********************************************************************
      OVERLAY (15,0)
      PROGRAM PRINT
      CHARACTER DMATRX,METHOD,CHOICE*6,PLTYPE*10,STRING*10
      CHARACTER ACT,SEN
      INTEGER P,PD
      REAL KO,K1,MM
      COMMON /B  1 / NOPT,IERR,IPLANT,IFLAG(40)
      COMMON /B  2 / N,M,P,ND,MD,PD
      COMMON /B  3 / GO(10,10)
      COMMON /B  4 / A(10,10),B(10,10),C(10,10)
      COMMON /B  5 / D(10,10)
      COMMON /B  6 / DMATRX,ACT,SEN
      COMMON /B  7A/ NA(10),AA(10,2,2),BA(10,2),CA(10,2)
      COMMON /B  7S/ NS(10),AS(10,2,2),BS(10,2),CS(10,2)
      COMMON /B  8 / ALPHA,EPSLON,GAMA(10,10)
      COMMON /B  8A/ METHOD
      COMMON /B  9 / KO(10,10),K1(10,10)
      COMMON /B 10 / NG,GN(10)
      COMMON /B 11 / MM(10,10)
      COMMON /B 12A/ XO(10),ZO(10),V(10)
      COMMON /B 12B/ ST,TT
      COMMON /B 13 / NT,YP(100,11,2)
      COMMON /B 13A/ UP(100,11,2)
      COMMON /B 13B/ ILIM,IMIN(10),IMAX(10),UMIN(10),UMAX(10)
      COMMON /B 14 / ICODE,NUM,ICOLMN(4),PLMAT(102,4)
      COMMON /B 14A/ CHOICE
      COMMON /B 15 / ICLM,NPAIRS,NDEPTH,IDEPTH(2)
      COMMON /B 16 / IENTRY,IFLTCN,LINES
      COMMON /B 17 / IPLOT
      DATA STRING/'1234567890'/
C ------------------------------------------------------------------
      IF (NOPT.EQ.100) GO TO 2100
      NOPT=NOPT-100
      IF (NOPT.EQ.6.OR.NOPT.EQ.7.OR.NOPT.EQ.8.OR.NOPT.EQ.9.OR.
     *NOPT.EQ.10.OR.NOPT.EQ.15.OR.NOPT.EQ.16.OR.NOPT.EQ.17.OR.
     *NOPT.EQ.19.OR.NOPT.EQ.20.OR.NOPT.EQ.28.OR.NOPT.EQ.29.OR.
     *NOPT.EQ.30) GO TO 1995
      IERR=NOPT
      IF (IFLAG(IERR).EQ.0) GO TO 9015
      GO TO (2101,2102,2102,2104,2105,1995,1995,1995,1995,
     *1995,2111,2112,2111,2114,1995,1995,1995,2118,1995,
     *1995,2121,2121,2123,2123,2123,2126,2127,1995,1995,
     *1995) NOPT
C ******************************************************************
C  OPTION #100
 2100 PRINT*, 'ALL 100-SERIES OPTIONS PRINT DATA VALUES...'
      PRINT*, '    FOR VALUES SET IN OPTION #1...USE OPTION #101'
      PRINT*, '    FOR VALUES SET IN OPTION #2...USE OPTION #102'
      PRINT*, 'ETC.'
      PRINT*, '    FOR PLOTTING SELECTIONS.......USE OPTION #130'
      GO TO 8015
C ******************************************************************
C  OPTION #101
 2101 PRINT*, 'THERE ARE ',M,' INPUTS AND ',P,' OUTPUTS'
      PRINT 4110
      CALL MATPR (GO,P,M)
      GO TO 8015
C ******************************************************************
C  OPTION #102 OR #103
 2102 PRINT*, 'THERE ARE ',N,' STATES, ',M,' INPUTS & ',P,' OUTPUTS'
      IF (NOPT.EQ.2) GO TO 8015
C ******************************************************************
 1800 PRINT*, 'ENTER...A,B,C,D...FOR PRINTOUT >'
      READ '(A)', PLTYPE
      IF (PLTYPE.EQ.'A') THEN
          PRINT 4075
          CALL MATPR (A,N,N)
          GO TO 8015
      ELSEIF (PLTYPE.EQ.'B') THEN
          PRINT 4080
```

```
              CALL MATPR (B,N,M)
              GO TO 8015
          ELSEIF (PLTYPE.EQ.'C') THEN
              PRINT 4095
              CALL MATPR (C,P,N)
              GO TO 8015
          ELSEIF (PLTYPE.EQ.'D') THEN
              IF (DMATRX.EQ.'N') THEN
                  PRINT '(/A)', ' THERE IS NO D MATRIX'
                  GO TO 8015
              ENDIF
              PRINT 4100
              CALL MATPR (D,P,M)
              GO TO 8015
          ELSE
              GO TO 1800
          ENDIF
C *************************************************************************
C  OPTION #104
 2104 DO 1805 K=1,M
          PRINT '(/)'
          PRINT*, 'ACTUATOR #',K,'...'
          PRINT 4115
          IF (NA(K).EQ.1) THEN
              PRINT 4120, AA(K,1,1),BA(K,1),CA(K,1)
              GO TO 1805
          ELSE
              PRINT 4125, (AA(K,1,J),J=1,2),BA(K,1),(CA(K,I),I=1,2)
              PRINT 4125, (AA(K,2,J),J=1,2),BA(K,2)
          ENDIF
 1805 CONTINUE
          GO TO 8015
C *************************************************************************
C  OPTION #105
 2105 DO 1810 K=1,P
          PRINT '(/)'
          PRINT*, 'SENSOR #',K,'...'
          PRINT 4115
          IF (NS(K).EQ.1) THEN
              PRINT 4120, AS(K,1,1),BS(K,1),CS(K,1)
              GO TO 1810
          ELSE
              PRINT 4125, (AS(K,1,J),J=1,2),BS(K,1),(CS(K,I),I=1,2)
              PRINT 4125, (AS(K,2,J),J=1,2),BS(K,2)
          ENDIF
 1810 CONTINUE
          GO TO 8015
C *************************************************************************
C  OPTIONS #111 AND #113
 2111 PRINT '(A,27X,1F10.5)',' ALPHA...',ALPHA
```

```
      PRINT '(A,1F10.5)',' SIGMA MATRIX MULTIPLIER, EPSILON...',EPSLUN
      GO TO 8015
C ****************************************************************
C  OPTION #112
2112 PRINT '(A/)', ' SIGMA WEIGHTING MATRIX...'
      CALL MATPR (GAMA,P,P)
      GO TO 8015
C ****************************************************************
C  OPTION #114
2114 IF (METHOD.EQ.'X') GO TO 1824
      IF (METHOD.EQ.'U') PLTYPE='UNKNOWN'
      IF (METHOD.EQ.'R') PLTYPE='REGULAR'
      IF (METHOD.EQ.'I') PLTYPE='IRREGULAR'
      PRINT*, 'CONTROL MATRICES ARE FOR PLANTS WHICH ARE ',PLTYPE
1824 PRINT '(/A/)', ' KO MATRIX...'
      CALL MATPR (KO,M,P)
      IF (ALPHA.EQ.1) THEN
          PRINT '(/A/)', ' K1 MATRIX IS IDENTICAL TO KO MATRIX'
          GO TO 8015
      ENDIF
      PRINT '(/A/)', ' K1 MATRIX...'
      CALL MATPR (K1,M,P)
      GO TO 8015
C ****************************************************************
C  OPTION #118
2118 PRINT '(A/)', ' MEASUREMENT MATRIX...'
      CALL MATPR (MM,P,N-M)
      GO TO 8015
C ****************************************************************
C  OPTIONS #121 AND #122
2121 PRINT '(A/)', ' X(0), INITIAL STATES...'
      PRINT 4130, (X0(I),I=1,N)
      PRINT '(/A/)', ' Z(0), INITIAL STATES...'
      PRINT 4130, (Z0(I),I=1,P)
      PRINT '(/A/)', ' INPUT COMMAND VECTOR, V (TRANSPOSED)...'
      PRINT 4130, (V(I),I=1,P)
      GO TO 8015
C ****************************************************************
C  OPTIONS #123 TO #125
2123 PRINT '(A,18X,5(F10.5))', ' SAMPLING TIMES...',(GN(I),I=1,NG)
      PRINT '(A,17X,F10.5)', ' SIMULATION TIME...',TT
      PRINT '(A,11X,F10.5)', ' CALCULATION STEP SIZE...',ST
      GO TO 8015
C ****************************************************************
C  OPTION #126
2126 PRINT*, 'ENTER...INPUT,OUTPUT...FOR DATA MATRIX >'
      READ '(A)',PLTYPE
      PRINT*, 'ENTER SAMPLING TIME OF INTEREST AS...1,2,3, ETC...>'
      READ*, IG
      IF (PLTYPE.EQ.'INPUT     ') THEN
```

```
         PRINT '(/A/)', ' INPUT DATA MATRIX...'
         PRINT '(5X,A,11X,10(A,12X))', 'TIME',(STRING(I:I),I=1,M)
         DO 1830 I=1,NT
1830     PRINT '(3X,10(1E10.4,3X))', (UP(I,J,IG),J=1,M+1)
         GO TO 8015
     ELSEIF (PLTYPE.EQ.'OUTPUT    ') THEN
         PRINT '(/A/)', ' OUTPUT DATA MATRIX...'
         PRINT '(5X,A,11X,10(A,12X))', 'TIME',(STRING(I:I),I=1,P)
         DO 1835 I=1,NT
1835     PRINT '(3X,10(1E10.4,3X))', (YP(I,J,IG),J=1,P+1)
     ELSE
         GO TO 2126
     ENDIF
     GO TO 8015
C *********************************************************************
C  OPTION #127
2127 DO 1840 I=1,M
     PRINT '  '
     PRINT*, 'CONTROL INPUT #',I
     PRINT*, '    MINIMUM VALUE IS ',UMIN(I)
1840 PRINT*, '    MAXIMUM VALUE IS ',UMAX(I)
     GO TO 8015
C *********************************************************************
C *********************************************************************
C  FORMAT STATEMENTS
C *********************************************************************
4075 FORMAT (/1X,'PLANT MATRIX A...'/)
4080 FORMAT (/1X,'PLANT MATRIX B...'/)
4095 FORMAT (/1X,'OUTPUT MATRIX C...'/)
4100 FORMAT (/1X,'DIRECT OUTPUT MATRIX D...'/)
4110 FORMAT (/1X,'MATRIX G(0)...'/)
4115 FORMAT (/'    MATRIX A',25X,'MATRIX B',10X,'    MATRIX C')
4120 FORMAT (1X,1F10.4,23X,1F10.4,11X,1F10.4)
4125 FORMAT (1X,1F10.4,1X,1F10.4,12X,1F10.4,11X,1F10.4,1X,1F10.4)
4130 FORMAT (3X,10(1E10.4,3X))
C --------------------------------------------------------------------
8015 IERR=0
     GO TO 9015
1995 IERR=43
     NOPT=NOPT+100
9015 CONTINUE
     END
*EOF
*EOR
```

Vita

Douglas S. Porter was born in Sheridan, Wyoming on 18 August 1950 and was raised in Edgemont, South Dakota. After completing his initial schooling in Edgemont, he attended South Dakota State University, Brookings, South Dakota, from September 1968 to December 1972. He graduated from SDSU with high honors in Electrical Engineering and was commissioned through the Air Force ROTC program from which he received a two-year scholarship.

After completing Undergraduate Pilot Training at Craig AFB, Alabama in January 1974 and Pilot Instructor Training at Randolph AFB, Texas, he served as a T-38 instructor at Moody AFB, Georgia and Reese AFB, Texas until September 1977. He was then assigned to the Tactical Fighter Wing at Elmendorf AFB, Alaska as an instructor, flight examiner and functional check flight pilot in the T-33 aircraft. He currently has a Senior Pilot rating with approximately 2,500 hours of jet aircraft time.

His additional educational experience includes graduation from the USAF Squadron Officer's School, Master of Business Administration courses from Valdosta State College, Valdosta, Georgia, and several hours in Engineering Management at the University of Alaska, Anchorage, Alaska.

He entered the School of Engineering, Air Force Institute of Technology in June 1980. His next assignment is with the Aerospace Systems Division at Wright-Patterson AFB, Ohio.

Permanent Home Address: 701 E Street, Box 654
Edgemont, South Dakota
57735

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/81D-48	2. GOVT ACCESSION NO. AD-A118 134	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DESIGN AND ANALYSIS OF A MULTIVARIABLE, DIGITAL CONTROLLER FOR THE A-7D DIGITAC II AIRCRAFT AND THE DEVELOPMENT OF AN INTERACTIVE COMPUTER DESIGN PROGRAM		5. TYPE OF REPORT & PERIOD COVERED MS Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Douglas S. Porter Capt		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Wright Aeronautical Laboratory (AFWAL/FIGL) Wright-Patterson AFB, Ohio 45433		12. REPORT DATE December, 1981 13. NUMBER OF PAGES 204
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17

[illegible]
LYNN E. WOLAVER
Dean for Research and
Professional Development

23 JUL

AIR FORCE INSTITUTE OF TECHNOLOGY (ATC)
WRIGHT-PATTERSON AFB, OH 45433

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Multivariable
Digital Control Laws
MULTI
Computer Design Package

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See Reverse

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

A multivariable, error-actuated digital tracking controller is developed for the Mach 0.18 flight condition of the A-7D Digitac II aircraft using a singular perturbation method. The design is accomplished and simulated after the development of an interactive computer program named MULTI. The complete design process is presented; a discussion of the robustness of the control law over a range of flight conditions and the effect of an aircraft flight control surface failure is also included. A more accurate aircraft model is required before further testing is accomplished to study the possibilities of other controller designs. The computer package is available to the engineering community.

FILME